(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 October 2001 (11.10.2001)

PCT

(10) International Publication Number
WO 01/75791 A2

(51) International Patent Classification[7]: G06N 3/12

(21) International Application Number: PCT/GB01/01517

(22) International Filing Date: 3 April 2001 (03.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0008291.7 4 April 2000 (04.04.2000) GB

(71) Applicant *(for all designated States except US)*: **THE UNIVERSITY OF WALES ABERYSTWYTH** [GB/GB]; Old College, King Street, Aberystwyth, Ceredigion SY23 2AX (GB).

(72) Inventors; and

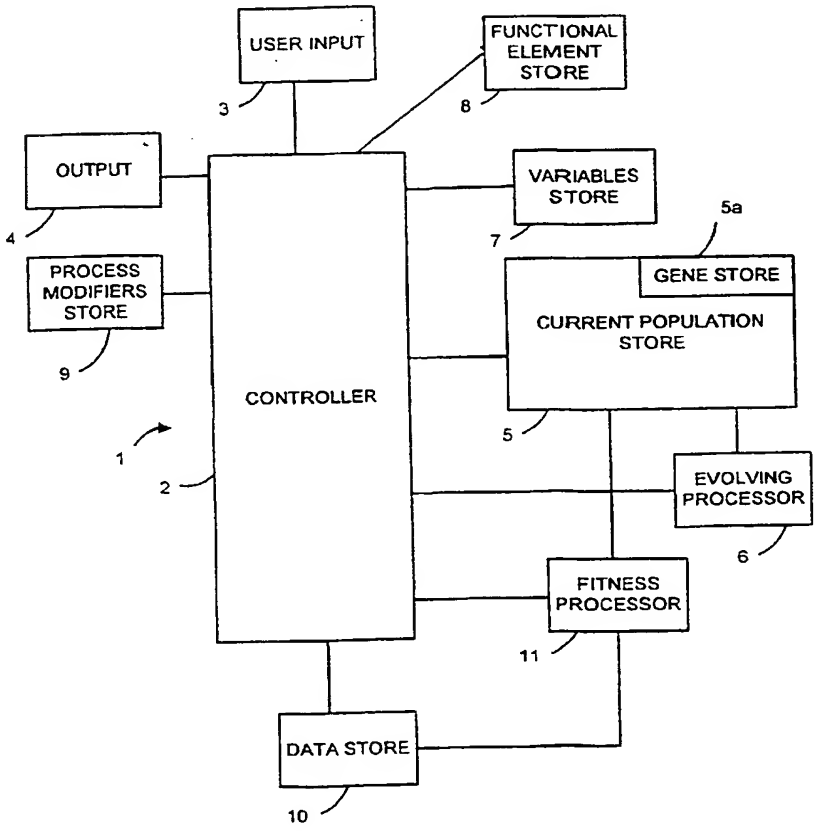
(75) Inventors/Applicants *(for US only)*: **KELL, Douglas, Bruce** [GB/GB]; Llechwedl Melyn, Pwgah, Aberystwyth, Ceredigion SY23 3NE (GB). **ROWLAND, Jeremy, John** [GB/GB]; Llwyn yr Eos, Capel Bangor, Aberystwyth, Ceredigion SY23 3LZ (GB). **GILBERT, Richard, James** [GB/GB]; Tan-y-Ffordd, Penuwch, Tregaron, Ceredigion SY25 6RA (GB).

(74) Agents: **BERESFORD, Keith, Denis, Lewis** et al.; Beresford & Co., 2-5 Warwick Court, High Holborn, London WC1R 5DH (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

*[Continued on next page]*

(54) Title: APPARATUS AND A METHOD FOR SOLVING PROBLEMS




(57) Abstract: A processor arrangement (2, 6, 11) is operable to provide an initial population of individuals each comprising a plurality of structural elements each comprising a plurality of functional components defining operations to be performed on at least one input to the functional component and then to: (i) activate each individual to cause the operations defined by the functional components of that individual to be performed to produce a result; (ii) determine from each result a fitness of the corresponding individual for providing a mechanism or program for providing a solution to the problem; (iii) produce a new generation of individuals by causing an evolutionary process to occur involving at least one structural element and at least one functional component; and then to repeat (i) to (iii) until either an individual of the current generation represents a suitable way, structure, mechanism or program for providing a solution to the problem or a predetermined number of generations have been evolved.

WO 01/75791 A2

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published:**

— *without international search report and to be republished upon receipt of that report*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

# APPARATUS AND A METHOD FOR SOLVING PROBLEMS

This invention relates to an apparatus and a method for solving problems, in particular for solving complex problems which may involve many variables.

Many complex problems, in particular problems involving multiple variables, cannot be solved easily by conventional means. Such problems are of high dimensionality, are often known as combinatorial optimisation problems, and include the so-called NP-complete problems in which it is not possible to determine the best answer without assessing all combinations of the available variables. Examples of such problems are the problem of seeking the best formulation of a cocktail of components for a chemical or biological product, for example a drug therapy, pattern recognition or classification, control and optimisation problems and problems in games theory such as the classical problem of the prisoner's dilemma.

Computer based techniques have been developed that aim to solve such complex problems by mimicking Darwinian natural selection or evolution. The genetic algorithm uses a population of potential solutions (individuals) each of which consists of a character string of, for example, numbers or binary digits representing parameters for determining a particular solution to the problem. Generally, the initial population is generated in a pseudo-random manner. The fitness of each of the individuals, that is how well the individual solves the

problem, is then evaluated in a manner dependent upon the problem to be solved. Thus, for example, where the problem relates to an aspect of games theory, each individual may consist of a games strategy for playing a particular game and a number of games may then be played using the strategy defined by each individual and then the results of those games assessed to see which individual strategy performs best. Where the problem to be solved is a search for the best cocktail or formulation of components or environmental conditions for forming a chemical or biological product, then the actual formulations may be prepared under laboratory conditions and then their suitability for the intended purpose tested under laboratory conditions to determine the fitness of the individuals of the population.

Once the fitness of each of the individuals of the initial population has been determined, then the population is caused to evolve with the probability of evolution of a particular individual being determined by its fitness. Various different evolutionary techniques or processes may be used. For example, an individual may simply be reproduced so that it survives unchanged to the next generation, an individual may be caused to mutate by introducing one or more pseudo-random changes and, as another possibility, components of two different individuals may be combined in a pseudo-random manner to produce a new individual (this technique is generally known as "crossover"). Each individual may experience one or more of these techniques, dependent on its

fitness. It is, however, sometimes beneficial to retain some individuals of "poor" fitness so as to maintain diversity in the population. If a particular individual is not subject to reproduction then it will not survive unchanged into the next generation. Generally, the least fit individuals will not survive into the next generation. Once the evolutionary processes has been completed and a new generation of individuals has been formed, then the processes of evaluation, fitness testing and evolution are repeated until either an optimum solution to the problem is obtained or the evolutionary process has proceeded through a pre-set number of generations. Where the latter is the case, then the most fit individual from the final generation may be taken as a suitable solution to the problem.

Problem solving by evolution using genetic algorithms has limitations in that the character strings are of fixed length and represent solutions to problems as an array of parameters so that, even though an optimum or suitable solution to the problem may be determined, the genetic algorithm does not necessarily provide any insight into the mechanism for solving the problem.

Another technique for solving problems by mimicking Darwinian evolution is so-called genetic programming which again starts with a pseudo-random population of individuals. However, in this case, the individuals do not represent potential solutions to the problem as arrays of parameters. Rather, each individual consists of a tree structure in which the leaves or terminals of

the tree represent input values which may be variables or numerical constants. Each terminal is connected via a branch to a node such that, generally, each node receives at least two inputs. Each node performs an operation on the received inputs and supplies an output on another branch which is connected to a further node which, like the preceding node, generally receives two input values and performs an operation on those two input values and supplies an output value to a further node. The tree will have a final node which is not connected to any further nodes but provides an output value from the tree.

The operations carried out by each node may be, for example, mathematical or logical operations such as computer operations or sub-routines. The fitness of each individual is evaluated on the basis of the output from the final or terminal node and, as in the case of the genetic algorithm technique, the initial population is subject to an evolutionary process wherein whether or not an individual is subject to at least one evolutionary process such as reproduction, crossover or mutation, depends upon the individual's determined fitness. In this genetic programming technique, mutation is effected by modifying the value or variable associated with a terminal or changing the operation performed by a node or sub-tree while crossover is effected by selecting two parents from the initial population, splitting the parent at pseudo-randomly selected nodes to form sub-trees and then combining or grafting together sub-trees from different parents. Genetic programming is described in

further detail in, for example, US patent numbers 4,935,877, US 5,343,554, US 5,390,282, US 5,742,738, US 5,867,397, US 5,742,738, US 5,148,513 and US 5,136,686, the whole contents of each of which are hereby incorporated by reference.

Genetic programming goes beyond the previously described genetic algorithm technique in that it provides the user not only with the optimum or otherwise suitable solution to the problem but also the structure of that solution. The difference between the genetic algorithm and genetic programming techniques can perhaps best be understood by way of an example. Thus, in order to use a genetic algorithm to find an explicit equation to fit a set of data, a user would be required to provide the general form of the equation (for example a fourth degree polynomial) and then the computer programmed with the genetic algorithm would find the appropriate parameters for that fourth degree polynomial. In contrast, where the genetic programming technique is used, it is not necessary for the user to provide the general form of the equation. Rather, in genetic programming, the individuals represent potential equations for fitting the data which equations are caused to evolve as discussed above in accordance with how well they fit the data. Genetic programming thus results in an individual representing the optimum equation for fitting the data or, if an optimum is not reached at the end of the pre-set number of generations, the equation providing the best fit evolved so far to the set of data. Genetic

programming thus enables solutions to problems to be found where little is known of the form of the solution.

Genetic programs do, however, have limitations in that the individuals of a population have a tree structure which may not be suitable or optimum for solving all types of problems. In addition, the tree structure is susceptible to bloat, and the fitness of each individual is judged (and consequently each individual evolves) independently of all others (which is not true in biology).

It is an aim of the present invention to provide an apparatus and method for enabling solving of complex problems by computer that uses an evolutionary process wherein a population of individuals is provided wherein each individual represents a potential way of solving the problem and consists of a plurality of rules or genes each in turn consisting of a plurality of sub-rules or functional elements and wherein not only the genes or rules may be allowed to evolve in accordance with the fitness but also the sub-rules in a manner which allows many different topologies for the individuals of the population and so allows greater flexibility.

In one aspect, the present invention provides apparatus for enabling the determination of an optimum or otherwise suitable way of solving a problem by evolving an initial population of individuals which each represent a potential way of solving the problem, which apparatus comprises means for defining an initial population of individuals each of which mimics a biological genome

comprising functional elements organised into genes which are themselves organised into one or more chromosomes constituting the genome and, wherein the chromosomes, genes and functional elements of an individual are all capable of evolution.

In one aspect, the present invention provides apparatus for enabling the development of a mechanism or algorithm for solving a problem by evolving an initial population of individuals each of which represent a potential mechanism or program for solving the problem, wherein each individual has a form analogous to a biological genome having functional elements organised into genes which are themselves organised into one or more chromosomes and wherein the apparatus comprises evolving means for causing evolution within the population in accordance with a determined fitness of the mechanism or program represented by an individual for solving the problem. The evolving means is arranged to operate at any one or more of the individual, chromosome, gene and function elements level so that evolution can occur in any one or more of the functional elements, genes and chromosomes or arrangement of chromosomes of an individual. The evolutionary processes may include reproduction, mutation in which a random change is effected in a functional element, gene or chromosome, and crossover which may involve swopping of functional elements between genes or swopping of genes between chromosomes, for example. Other evolutionary processes mimicking transposition, linkage, cross-regulation,

operons, polyploidy, dominance, recession, deletion, insertion and gene transfer may also be used.

Embodiments of the present invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 shows a functional block diagram of apparatus embodying the invention;

Figure 2 shows a structural block diagram of a computer system that, when programmed, may provide the apparatus shown in Figure 1;

Figure 3 shows a top level flow chart for illustrating a method embodying the invention;

Figure 4a shows diagrammatically the structure of a functional element of an individual of a population provided by apparatus embodying the present invention;

Figure 4b shows diagrammatically the components of a descriptor of the functional element shown in Figure 4a;

Figure 4c shows diagrammatically the components of a gene of an individual;

Figure 4d shows diagrammatically the components of a descriptor of the gene shown in Figure 4c;

Figure 5 shows diagrammatically the structure of an individual of a population provided by apparatus embodying the invention;

Figure 6 shows diagrammatically the components of a descriptor of an individual;

Figure 7 shows diagrammatically the components of a descriptor of a chromosome of the individual shown in Figure 5;

Figure 8 shows diagrammatically an example of a specific functional element;

Figure 9 shows diagrammatically an example of a specific gene or rule;

Figure 10 shows a flow chart for explaining the generation of an initial population of individuals using apparatus embodying the invention;

Figure 11 shows steps carried out in evolving a population to produce the next generation;

Figure 12a shows steps carried out in effecting a crossover evolutionary process;

Figure 12b shows diagrammatically the effect of performing a crossover evolutionary process on two structural elements that may be functional elements, genes or chromosomes;

Figure 13a shows steps carried out in a mutation evolutionary process;

Figure 13b shows diagrammatically the effect of carrying out a random mutation on a functional element, gene or chromosome;

Figure 14a shows steps carried out in a transposition evolutionary process;

Figure 14b shows the effect of carrying out a transposition evolutionary process on a functional element, gene or chromosome;

Figure 14c shows the effect of carrying out the transposition evolutionary process on a functional element, gene or chromosome wherein individual components within the functional element, gene or chromosome are linked;

Figure 15a shows steps carried out in a deletion evolutionary process;

Figure 15b illustrates the effect of carrying out a deletion evolutionary process on a functional element, gene or chromosome;

Figure 16a shows steps carried out in an addition evolutionary process;

Figure 16b shows the effect of carrying out an addition evolutionary process on a functional element, gene or chromosome;

Figure 17 shows the effect of a duplication evolutionary process;

Figures 18a and 18b show the effect of an evolutionary process using recession and dominance, respectively, on a functional element, gene or chromosome;

Figure 19 shows a flow chart for enabling complexity to be taken into account in determining the fitness of an individual;

Figure 20 shows in greater detail a way of taking complexity into account;

Figure 21 shows a flow chart illustrating an evolutionary process using demes;

Figure 22 shows a flow chart illustrating in greater detail the step of modification of a deme population shown in Figure 21; and

Figures 23a to 23d show various different topologies that may be adopted by functional elements, genes or chromosomes of an individual provided by apparatus embodying the present invention.

Figure 1 shows a functional block diagram of computer apparatus (1) embodying the invention for enabling the development of a way, structure, mechanism or computer program for obtaining a solution to a complex problem by subjecting a pseudo-randomly generated initial population of individuals, each of which represents a potential way, structure, mechanism or program for solving the problem to evolutionary processes with the aim of producing an individual way, structure, mechanism or program that provides the optimum or otherwise suitable way, structure, mechanism or program for solving the problem.

The functional elements of the apparatus comprise a controller (2) which controls the overall process carried out by the apparatus, a user input device or devices (3) for enabling a user to input data and instructions to the controller (2), an output device or devices (4) for enabling the controller to supply a user with the results of a process carried out by the apparatus, and a current population store (5) that stores information defining the current population of individuals. The current population store (5) may include a separate gene store

(5a) as will be described in detail below. The apparatus also includes an evolving processor (6) that, under control of the controller (2), causes evolution of individual members of the current population as will be described below. The apparatus also includes a variable store (7) for storing variables that need to be accessible by the current population ("globally accessible variables"). A separate functional element store (8) may be provided to store individual functional elements that may be used for building individual members of the population. As shown in Figure 1, the apparatus also includes a process modifiers store (9) for storing information that, as will be explained below, may be used to modify the behaviour of one or more individuals of the population. The apparatus also includes a fitness processor (11) for determining the fitness of individual members of a population and a data store (10).

The apparatus shown in Figure 1 may be implemented by suitable programming of a conventional computer system. Figure 2 shows a block diagram of a typical computer system (100) which comprises a processor unit (101) and associated memory (RAM and/or ROM) (12) a mass storage device (110) such as a hard disc drive, a removable media storage device (13) for receiving a removable media disc (14) such as a floppy disc or an optical storage disc such as a CD ROM, a communications interface such as a MODEM or network interface (15) for enabling communication with other computer systems, via a network or, for example, the worldwide web user input

devices such as a keyboard (16) and a mouse (17) and output devices such as a display (18) and a printer (19) for providing a user with information.

Processor implementable instructions for causing the computer system (100) shown in Figure 2 to become configured to provide the apparatus (1) shown in Figure 1 may be supplied or downloaded to the processor unit (101) by any one or a combination of the following mechanisms, namely: input of processor implementable instructions by a user using the keyboard (16), as a signal (15a) via the communications interface (15), and on a storage medium (14) receivable in the removable media storage device (13). The processor implementable instructions and/or data cause the processor unit (101) and associated memory (12) to become configured to carry out the functions illustrated by the controller (2), evolving processor (6) and fitness processor (11) shown in Figure 1. The current population store (5), variables store (7), functional element store (8) and process modifiers store (9) may be provided by the mass storage device (110) in conjunction with the working memory (12) of the processor unit (101) and possibly also a storage medium (14) received in the removable media storage device (13). Although Figure 2 shows a computer system having a single processor unit (101), for reasons which will become evident from the following, the present invention is well adapted to a parallel processing arrangement and accordingly the computer system (2) may comprise a parallel computing system which, as is well

known in the art, will generally have a shared memory resource.

Figure 3 shows a top level flow chart for illustrating the process carried out by the apparatus (1) for enabling development of a way, structure, mechanism or computer program for attaining a solution to a problem by evolving a population of potential ways, structures, mechanisms or programs for solving that problem.

Initially, at step S1 the controller (2) defines an initial population of individuals and stores that population in the current population store (5). The initial population may be generated using conventional pseudo-random techniques using functional elements building blocks stored in the functional elements store (8) and variables stored in the variable store (7) or may be generated using data supplied by a user, or combination of these.

The controller (2) then activates each individual of the current population to cause it to carry out the operations specified by that individual and stores the result for each for each individual in the data store (10) of Figure 1 (step S2). The controller (2) then causes the fitness processor (11) to test the results of carrying out the operations specified by each individual to assess the fitness of that individual, that is how close the solution resulting from the operations carried out by that individual is to an optimum or best fit solution to the problem (step S3).

The tests carried out at step S3 will depend upon the problem being addressed. For example, where the problem is one of games theory and the individuals represent different game strategies, then a series of games may be physically played or simulated by the controller (2) and the individual's ranked in accordance with their average pay-off so that the individual providing the best average pay-off is determined to have the best fitness, i.e. the best strategy (of those represented in the current population) for playing that game, while the individual providing the worst average pay-off is determined to have the worst fitness. Where the aim is to provide a model or equation representing the best or otherwise suitable fit to a set of data, then a validation procedure may be used in which the resulting model or equation is tested against data that was not used in the generation of that model or equation. This guards against the problem of "overfitting" where the model or equation represents the training data so well that it fits the noise as well as the training data and is therefore incapable of generalising for the interpretation of new unseen data whose noise characteristics are invariably different. There are many different validation procedures. However all of them rely on checking the equation or model produced against data that has not been used in forming the model or equation that is being validated. Thus, the individuals may be tested to see which individual provides the equation which best fits not only the portion of the set

of data that was made available to the controller (2) on initiation of the process but also other data in the same set. Generally, in modelling, three sets of data are used: one set for training, one set for determining the point at which the model or equation begins to fit the noise and a third set that is completely unseen until it is used as an independent check of the capabilities of the equation or model that has been produced. The standard deviation of a line representing the equation from the actual data may be used as a test of fitness.

As described at pages 94 to 98 of the book "Genetic Programming" by John R Koza published by the MIT Press (ISBN 0-262-11170-5) in 1998 there are various different types of fitness determination such as "raw fitness", "standardised fitness", "adjusted fitness" and "normalised fitness" and the type of fitness determination used may be selected by the user or the controller (2) in accordance with the particular problem being addressed. For example, a normalised fitness may be used so that the fitness of each individual is expressed as a proportion of the average fitness. In this case the fitness ranges between 0 and 1. Once the fitness of each individual of the initial population has been determined at step S3, the controller (2) checks at step S4 whether any of the individuals meets the desired fitness goal, that is whether any of the individuals actually provides a way, structure, mechanism or computer program for obtaining the optimum or otherwise suitable solution to the problem. If the answer at step S4 is

yes, then the controller (2) causes the output device (4), generally the display (18) shown in Figure 2, to advise the user at step S5 that the fitness goal has been reached and to provide the user with details of the individual meeting the fitness goal. The process is then at an end.

It is, however, extremely unlikely that the initial population of individuals will contain the optimum mechanism or computer program for solving the problem. Accordingly, generally the answer at step S4 will be no and the controller (2) will then check at step S6 whether this is the nth generation, that is whether the population has already been allowed to evolve to a predetermined number of generations that may be set by the user or may be a default set by the controller (2). In the present case, the answer at step S6 will be no because no evolution has yet taken place. Accordingly, at step S7, the controller (2) causes the evolving processor (6) to cause individuals having a certain fitness or fitness range in the current population to evolve in a certain manner to produce a new population which the controller (2) then causes to be stored at the current population in the current population store (5).

The controller (2) then repeats step S2 by activating each individual of the new population or generation to carry out the operations specified by that individual and stores the results in the results data store (10). The fitness data processor (11) then tests the results and allocates a fitness to each individual at

step S3 as described above. This information is stored in the data store (10). The fitness data processor (11) then determines if the fitness goal has been achieved at step S4 and if so advises the user at step S5 as set out above. If not, then the controller (2) determines whether this is the nth generation at step S6 and if not repeats steps S7 to S6 until the answer at step S6 is yes. When the answer at step S6 is yes, the controller (2) determines that the pre-set number of generations has been evolved without producing an individual that provides the optimum fit and accordingly advises the user at step S8 of the fittest member of the current population, that is the best result so far.

In accordance with the present invention, the apparatus (1) provides the initial and subsequent individuals of the population in a form analogous to a natural biological genome as will now be explained in greater detail with respect to Figures 4a to 9.

Thus, each individual is in the form of a hierarchical structure comprising functional elements or sub-rules organised linearly into genes or rules which are themselves organised into one or more chromosomes or groups of rules which together form an individual or set of groups of rules.

Figure 4a shows diagrammatically the structure of a functional element or sub-rule (20). The functional element (20) comprises an extensible descriptor block (21) that determines the characteristics of the functional element and also includes or has a pointer to

an executable operational function to be carried out by the functional element. The descriptor (21) may be followed, depending upon the particular operator function to be carried out by that functional element, by one or more arguments. In the example shown in Figure 4a, the functional element (20) has a series of arguments A1, A2.....AN.

Figure 4b illustrates in greater detail the structure of the extensible descriptor block (21) of a functional element (20). The descriptor block (21) consists of a first section (21a) that contains information determining the type of the functional element, a second section (21b) that contains information determining the function or operation to be carried out by the functional element, a third section (21c) which indicates the number of arguments that follow the descriptor and sections ($21b_1$ to $21d_n$) each corresponding to a respective different one of the arguments and containing information specifying the type of the argument.

The descriptor (21) may also include a counter (21e) which keeps a record of the number of times that the function or operation represented by the functional element has been invoked. This enables, as will be explained below, problems such as bloat and undue complexity to be avoided or at least reduced.

Figure 4c shows diagrammatically the structure of a gene or rule 30. In a manner similar to the functional element (20), a gene (30) contains a descriptor (31)

followed by one or more functional elements. The gene (30) shown in Figure 4c has functional elements E1, E2, EN. As shown in Figure 4d, the descriptor (31a) of a gene contains an identification (ID) section (31) containing information enabling the controller (2) and evolving processor (6) to identify the particular gene, followed by a section (31b) containing information indicating the length of the gene, that is the number of functional elements contained in the gene, a section (31c) containing information indicating the functional element which should be used as a starting point when executing the operations defined by the gene and an attenuator section (31d) for containing information that may modify the output or result of the gene in accordance with, for example, external conditions as will be described in greater detail below. The descriptor (31) may also include a local variables section (31e) identifying variables specific to the gene.

Figure 5 shows a diagrammatic representation of part of an individual genome. As shown in Figure 5, the individual or genome consists of a descriptor block (51) followed by a list of chromosomes, chromosomes C1, C2, C3,.....CN being shown in Figure 5. Each chromosome (40) consists of a descriptor block (41) followed by a list of genes. Figure 5 shows diagrammatically only the structure of the chromosome C1 which is represented as having a sequence of genes G1, G2, G3.....GN. It will, of course, be appreciated that each of the individual chromosomes (there may be 1, 2 or more) of the individual

or genome may have a different number of genes. Each of the genes of a chromosome (40) consists of one or more functional elements and has the structure shown diagrammatically in Figure 4 and represented for the gene G1 in Figure 5. Thus, the gene G1 consists of a descriptor (31) followed by functional elements E1, E2, ...EN in this example. Again as described above, each functional element of each gene has a descriptor (21) which may be followed by one or more arguments, depending on the operational function carried out by the functional elements. A functional element E1 of the gene G1 is shown in Figure 5 as having two arguments A1 and A2.

Figure 6 shows diagrammatically a descriptor (50) of an individual. As shown, the descriptor includes an ID section (51a) identifying the individual to the controller (2), a section (51b) containing information identifying the number of chromosomes in the genome, a section (51c) indicating the chromosome that provides the start point for executing the series of functions defined by the genome and an attenuator section (51d) that contains information that may attenuate or modify the output of the genome. The information in the attenuator section may be, for example, influenced by the general environment of the population or the presence or absence of particular global variables in the variables store (7). The individual descriptor (51) may also include a section (51e) storing variables that are local or specific to that particular individual and are used only for execution of that particular individual.

Figure 7 illustrates diagrammatically the structure of the descriptor (40) of a chromosome. This again includes an ID section (41a) for identifying the chromosome followed by a section (41b) containing information identifying the number of genes in the chromosome, a section (41c) identifying the gene from which execution of the chromosome should be started and an attenuator section (41d) for containing information that may attenuate or modify the result of execution of the chromosome. Again this information may be dependent on the environment of the chromosome, for example the other chromosomes in the same individual or the existence or absence of features in other individuals or global variables. The chromosome descriptor (40) may also include a local variable section (41e) identifying variables specific to that particular chromosome within the individual. It will, of course, be appreciated that, if present, sections (51e and 41e) and (31e) of the individual and chromosome and gene descriptors may consist simply of pointers to information in the variable store (7).

An individual may have one or more chromosomes; where the individual has a single chromosome then it mimics a typical prokaryotic organism while where the individual has multiple chromosomes then it mimics a prokaryotic organism containing plasmids. The chromosome complement may be duplicated to represent a diploid chromosome as is present in most eukaryotes or may be a

haploid chromosome arrangement as in the sex chromosomes of mammals.

As explained above, there may be a number of different types of functional elements. One example of a type of functional element is a numeric or number functional element, that is a functional element that is arranged to operate on numbers. The function carried out by such a functional element may be a numeric operation such as: add, subtract, multiply, divide etc. The functional element may also be a logic element capable of carrying out logic operations on binary numbers such as AND, OR, NAND, EXOR. The functional element may also be a logic or conditional statement of the type used in computer programs such as, for example, "If then, else" or "If else" or "Go to". The arguments may be variables or constraints, for example variables or constraints local to the gene, chromosome or individual or global variables, user defined variables or constraints, or outputs from other genes.

As will be appreciated, when activated, a functional element (20) invokes or activates the operator function specified in its descriptor block (21), takes the values specified in any associated arguments (arguments A1 to AN in Figure 4) and carries out the operation specified by the operator function in accordance with those arguments and returns a value that may be utilised by the gene in accordance with its remaining functional elements. The genes perform a similar function in relation to the

chromosome and the chromosomes in relation to the individuals.

In the examples described above, the functional element has a strictly defined type so that, for example, a functional element which performs the numeric or arithmetical "add" operation may accept only two numbers as arguments and provide only a number as its return value. The descriptor block of the functional elements may further specify that the number is either an integer or a floating point value. Causing the functional elements to have strictly defined types ensures that values can only be passed between functional elements of compatible types so that, for example, a text string cannot be passed to a functional element whose function is to carry out the numeric or arithmetical "add" function while a functional element whose function is to carry out a Boolean logic operation such as "AND", "OR", etc. will normally specify as its type binary numbers and so will only accept binary inputs. Other functional elements may be of different types, for example, a functional element may be arranged to accept text or character strings and, for example, to carry out an add operation to concatenate those strings. Specifying the type of a functional element provides two benefits, it reduces the likelihood of logically inconsistent solutions and may enable a clearer mechanistic explanation of a solution.

The functional elements discussed above provide a single return value. This need, however, not necessarily

be the case and the functional element may be such as to provide two or more return values which need not necessarily be of the same type. For example, a functional element encoding a logical "if" function may return three values, a Boolean value (e.g. true), a fuzzy logic class (e.g. very likely) and a probability (e.g. 0.96). In this case, the function that invoked the functional element, namely the gene in this case, may make use of whichever one or combination of return values it requires, dependent upon the further processing to be carried out by that gene.

Figure 8 illustrates one specific example of a functional element (20). This functional element is of a numeric type and its function is to add the two arguments one of which represents a variable $I_3$ and the other of which represents a number, in this case 0.287. Accordingly, the output of the functional element (20) shown in Figure 8 is $I_3 + 0.287$. As discussed above, the variable $I_3$ may be a global variable that is accessible by all of the individuals in a population, or may be a variable that is local to the gene, chromosome or individual or even the output from another gene that need not necessarily be in the same individual.

Figure 9 shows a specific example of a gene that decodes to the equation shown in Figure 9 beneath the gene. In this case, the gene has a length of six, that is there are six functional elements E1 to E6, the functional element at which execution of the gene starts is E4 and the gene has an expression or attenuator value

of 0.8 so that the result of executing the functions specified by the gene is multiplied by 0.8 to provide the output. Although the gene is represented as a linear array of functional elements, as shown in Figure 9 the flow of operations within the gene may be non-linear. The functional elements may include recursive calls, loops or conditional branching and the input variables for the gene, ($I_4$, $I_2$ and $I_7$ in the examples shown in Figure 9), may obtain their values from a user provided data set from a local variable set specific to that gene, chromosome or individual or from a pool of global variables stored in the variable store (7) and accessible to all members of the population.

The expression or attenuator in the descriptor block of the gene may be controlled or modified by functional elements from within the same gene or from other genes in the same chromosome, for example, allowing for cross-regulation of genes. As a further possibility, the expression or attenuation factor may be controlled or modified by the presence or absence of specific data in the process modified store (9) which thus functions in a manner analogous to metabolites in biological processes.

As will be appreciated from the above, genes may encode mathematical formulae (for example output = sin (x) + 4y), logical rules (for example if $x \geq (y/3.7)$ then true else false), data mining expressions (for example if (agent [1] said yes) and (agent [3] > agent [5] then return record [7]) or other representational form. As another possibility, genes may encode or represent

particular combinations of electric circuit elements. As will be appreciated from the above, a chromosome consists of a group of genes or rules which when the chromosome is activated will be executed in the order determined by the chromosome structure or content (that is the rules forming the genes). Similarly the chromosomes within an individual will be executed in the order determined by the individual or genome structure or content.

Figure 10 is a flowchart showing in greater detail the step S1 in Figure 3 of defining the initial population of individuals. At step S9, the controller (2) determines the parameters for the evolutionary process to be carried out. These parameters will include: the set of functional elements that can be used; any limitations on the structure and composition of genes, chromosomes and individuals, for examples limitations on the length of any of these; the variables that can be used by the process including those that can be accessed by all individuals and those that may be specific to a particular individual, chromosome or gene; the fitness criteria and how fitness is to be evaluated, for example which of the fitness evaluation methods described above may be used; the number of generations of evolution before the process terminates; the types of evolution possible and probability factors which determine the probability of evolution occurring to a particular individual in accordance with its fitness. All of this information may be in the form of default values or information stored in the mass storage device

(110). Alternatively or additionally, a user may specify any one or more of these parameters. For example, where a user has carried out the evolutionary process on similar sets of data for similar reasons (for example training procedures), then the user may make use of the experience gained from those earlier procedures to define certain parameters or values in a way that he, from experience, considers should facilitate the evolutionary process and enable a desired result to be achieved in a small number of generations. The mass storage device (110) may be arranged to contain different sets of functional elements and parameters and variables for different types of problems, for example different sets of functional elements, parameters and variables for different general types of problems so that a user can select a set of functional elements, parameters and variables that are specifically well adapted for the general type of problem with which he is concerned. For example, different sets of functional elements, and variables may be provided for games theory problems and for control processes or formulation problems. Thus, at step S9, the controller (2) may cause the output device (4) to request the user (3) to indicate whether the user wishes to define particular parameters for the process or whether the user wishes to use the default or a particular set of default values. Once the user has answered this question using the keyboard (16) or mouse (17), then the controller (9) determines the parameters to be used for the process at step S9 and then generates

a set of genes from the determined set of functional elements at step S10 using a conventional pseudo-random number technique. The genes may be stored in the data store (10) or in a specific gene store (5a) associated with the current population store (5). The controller (2) then generates from the genes, again in a pseudo-random manner, a set of chromosomes at step S11 and then uses the generated chromosomes to generate, again in a pseudo-random manner, an initial population of individuals at step S12. This initial population is then stored in the current population store (5).

The advantage of providing a separate gene store (5a) is that each individual chromosome does not need to include all the information for the genes but may simply contain a pointer to the relevant gene in the gene store (5a) so that the or each chromosome of an individual effectively comprises a list indicating which of the genes in the globally accessible gene store are present in its genome. By analogy to the biological world where there may be several minor variants of the "same" gene (known as alleles) the gene store need not necessarily store each minor variant of a gene but may store full information for a base gene and for each gene which constitutes a minor variation of that base gene simply the difference information, so reducing the amount of information that needs to be stored.

Once the initial population has been provided, then the controller (2) causes the operations specified by each individual to be executed and the results to be

stored in the data store (10) as set out at step S2 in Figure 3 and the fitness processor then determines the fitness of each individual at step S3 in Figure 3. Assuming that the fitness goal is not immediately achieved and that this is not the final generation, then the controller (2) causes the evolving processor (6) to evolve individuals in the population having a certain fitness to produce a new population.

Whether or not evolution occurs for a particular individual may be determined solely by the specific fitness value associated with that individual so that evolution will occur if an individual has a specific fitness or a fitness that lies within a specified range. As another possibility, whether or not evolution of a particular individual occurs may be determined on a probability basis which may be related to, for example, the fitness value alone or in combination with other factors such as external influences, for example the presence or absence of particular features or information in the process modifiers store (9) or the presence or absence of specific genes or chromosomes in other individuals within the population. In addition, there may be different probabilities for evolution at the different levels within an individual so that, for example, the probability of evolution at the functional element level may be different from the probability of evolution at the gene level which may again be different from the probability of evolution at the chromosome level. Whether evolution occurs at one or more of these levels

may be determined by the fitness value or by the fitness value in combination with external influences as discussed above. The circumstances under which evolution will occur for a particular individual and the level within that individual at which evolution may occur may be randomly defined, or be defined by pre-set values which may be default values or may be values supplied by a user or as the result of training programs carried out on similar data sets.

Figure 11 shows a flow chart illustrating the basic process of evolution for a particular individual already selected for evolution. At step S13, the evolving process (6) determines the level of evolution, that is the evolving processor (6) determines whether evolution is to occur at the individual level, the chromosome level, the gene level, the functional element or a combination of any two or all of these. As mentioned above, the level or levels at which evolution may occur may depend solely on the fitness value and probabilities allocated to the different levels of evolution or additionally on the presence of process modifiers in the process modifier store (9) or the presence or absence of certain variables in the variable store (7) or the presence or absence of certain genes in that individual or other members of the population. Once the evolving processor (6) has determined the level at which evolution should occur, then the evolving processor (6) determines at step S14 the type of evolution to be effected. Again, the type of evolution to be effected may be determined by

the fitness value associated with the individual or may be selected at random or may be determined by external influences such as the presence or absence of other genes or process modifiers or metabolites in the process modifier store (9). As another possibility, the level and type of evolution may be chosen in a pseudo-random manner once an individual has been selected for evolution. Once the level and type of evolution have been determined, then the evolving processor (6) causes the determined type of evolution to be carried out at step S15 and at step S16 stores the result of the evolution in the current population store (5) in place of the individual or individuals in which the evolution process was carried out, so producing a new generation.

As mentioned above, there are many different forms of evolution process. When an evolution process such as crossover is to be performed, then crossover may occur within an individual or between individuals, where the latter is the case, then the evolving processor (6) may select the individuals to be subject to the crossover evolutionary process randomly from a group of individuals having a certain fitness or fitness range. Of course, other criteria may be used to select the individuals for crossover, for example the presence or absence of certain genes within the individuals.

Examples of the evolutionary processes that may be carried out by the evolving processor (6) will be described below with reference to Figures 12a to 18b. These evolutionary processes may, as described above, be

carried out at functional element, gene, chromosome or individual level. Accordingly, in the following description of Figures 12a to 18b, the phrase "structural element" should be understood to mean either a functional element, a gene, a chromosome or an individual while the word "component" should be understood to mean a component of that structural element so where the structural element is a functional element, the components are arguments, where the structural element is a gene, the components are functional elements, where the structural element is a chromosome, the components are genes and where the structural element is an individual, the components are chromosomes. Where an evolutionary process is occurring between two structural elements (as in the case of crossover, for example), then it should be understood that the two structural elements will be of the same type (i.e. both functional elements, both genes, both chromosomes or both individuals).

Figure 12a illustrates the basic steps carried out by the evolving processor (6) during a crossover evolutionary process assuming that the structural elements to be subjected to the crossover process have already been determined as described above. As step S17, the evolving processor determines the crossover point or points in each structural element. Generally, these will be selected in a pseudo-random manner. Where the structural element is a functional element, then the crossover point will define a separation between the arguments, while where the structural element is a gene,

the crossover point will define a separation between the functional element of that gene and so on. Figure 12a shows a first structural element (which may be a functional element, gene, chromosome or individual) (100a) having a descriptor (1) and components (x11, x12 and X13) and a second structural element (100b) having a descriptor (2) and components (x21, x22 and x23). In this case, the determined crossover point is indicated by the dashed line X.

Once the crossover point has been determined, the evolving processor (6) splits the two structural elements at the crossover point at step S18 in Figure 12a and then, at step S19 in Figure 12a, performs the crossover evolutionary process by, in this example, combining the components (x11) of structural element 1a on one side of the crossover point X in the structural element (100a) with the components (x22 and x23) on the other side of the crossover point from the structural element (100b) and combining the component (x21) from one side of the crossover point X of structural element (100b) with the components (x12 and x12) from the other side of the crossover point from structural element (100a). The crossover evolutionary process results in two new structural elements with the structural element (100c) having components (x11, x22 and x23) and a descriptor (1a) modified to reflect the change in the component. The other structural element (100d) resulting from the crossover operation has a descriptor (2a) and components (x21, x12 and x13). Thus, the crossover evolutionary

process results in two new structural elements that replace the original structural elements (100a and 100b) in the next generation.

Figure 13a shows a flow chart illustrating the basic steps carried out by the evolving processor (6) to mutate a structural element. At step S20, the evolving processor (6) determines the feature to be mutated, again this may be determined on a pseudo-random basis or may be influenced by external factors such as the presence or absence of specific genes, variables or process modifiers ("metabolites"). Once the feature to be mutated has been determined at step S20, then the evolving processor (6) mutates that feature and stores the mutated structural element in place of the previous structural element at step S21. Figure 13b shows a structural element (100e) having a descriptor (1) and components (x1, x2 and x3) being mutated into a structural element (100'e) having a descriptor (1a) and components (x1, x'2 and x'3). In this example, therefore, the component (x2) has been mutated. The mutation may be carried out in a pseudo-random manner so that, for example, where the component is a variable forming an argument of a functional element, that variable may be randomly swopped for another variable. As another possibility a constant forming an argument may be randomly changed.

Figures 14a and 14b illustrate another evolutionary process that the evolving processor (6) may carry out, namely transposition. In this case, the evolving processor (6) determines components within the same

structural element (100f) (Figure 14b) to be swopped. Again these may be determined in a pseudo-random manner. In the examples shown in Figure 14b, the components to be swopped are components (x2 and x3). Once the components to be swopped have been determined, then the evolving processor (6) swops the elements at step S23 and stores the new structural element (100'f) in place of the original structural element (100f). As can be seen from Figure 14b, the new structural element (100'f) differs from the old structural element (100f) simply by the swapping in position of the components x3 and x2.

Components within a structural element may be linked together, for example genes within a chromosome may be linked together, so that, for example, when a crossover or transposition process is selected and one of the components selected for crossover or transposition is linked to an unselected component, that unselected component moves with the selected component as if the two were a single component. Figure 14c illustrates the case where a structural element (100g) has linked components (x3 and x4) and transposition between components (x2 and x3) has been selected. In this case, when the evolving processor (6) carries out the transposition process and determines that the component (x3 and x4) are linked, the components (x3 and x4) move together so that, after the transposition process, a structural element (100'g) is produced in which the components are ordered as follows: (x1, x3, x4 and x2). Linking of components such as genes together can be used to reduce or avoid the possibility

of disruptive crossovers or transpositions which may otherwise separate functionally interdependent components.

Figures 15a and 15b illustrate the process of deletion of a component from a structural element. In this case, the evolving processor (6) determines, usually in a pseudo-random manner, the component to be deleted at step S24 and then deletes that component at step S25 and stores the structural element with that component deleted in the current population store at step S25. Figure 15b illustrates a structural element (100h) having a descriptor (1) and components (x1 to x5) and the modified structural element (100'h) having a descriptor (1a) and components (x1, x3, x4 and x5) which results from the evolving processor having determined that component (x2) should be deleted. Figures 16a and 16b illustrate the complementary process of adding a component to a structural element. At step S26, the evolving processor (6) determines the component to be added and the location within the structural element at which the component is to be added. Both of these may be determined in pseudo-random manner. Figure 16b shows a structural element (100k) having a descriptor (1) and components (x1 to x4) where the position at which an additional component is to be added has been selected as between components (x2 and x3). At step S27 the evolving processor (6) adds the selected component to the structural element and stores the modified structural element (100'k) in the current population store in place of the structural element

(100k). As can be seen from Figure 16b, the structural element (100k) has an additional component (x5) between components (x2 and x3).

Figure 17 illustrates the relatively straightforward evolutionary process of duplication or asexual reproduction in which a structural element (100m) selected for asexual reproduction or duplication is copied into the next generation without modification. The duplication process may be carried out in accordance with the fitness values and/or pseudo-random manner so that one, two or more duplicates of a particular individual may be produced. As shown in Figure 17 two duplicates are produced.

Because the individuals of the population mimic natural genomes, many of the other evolutionary processes that occur in Darwinian natural selection may also be used. For example, chromosomes may be doubled-up and gene sequences within individuals may be doubled-up or randomly reordered. Also, genes may be designed to have recessive or dominant properties so that, for example, where a gene has a recessive property and an offspring is being generated from two parent chromosomes (100n) and (100p) as shown in Figure 18a, then any gene in a parent that has a recessive property will not be reproduced in the offspring unless that gene is present in both parents. Thus, in the example shown in Figure 18a, the gene (G5) is present only in the parent (100p) and, because it is a recessive gene, it is not present in the offspring (100q). Of course, in this example, it does

not matter whether any of the genes (G1 to G4) are recessive because they are present in both parents.

Figure 18 illustrates the complementary process where the same two genes (100n and 100p) are involved but in this case the gene (G5) is dominant rather than recessive and so is present in the offspring (100r).

To avoid the possibility of the non-linear nature of the structural elements resulting in infinite loops whereby the flow of execution becomes trapped in a cyclic fashion, a computational penalty may be employed that effectively decreases the fitness of the structural element if it shows a tendency to become stuck in an infinite loop. One way of achieving this is to use the counter in the descriptor of the structural element as shown in Figure 4b for the descriptor (21) as a functional element (20), and to set this counter at zero at the start of a process and to increment the counter each time the structural element is invoked. Where this is the case, then the evolutionary processes carried out by the evolving processor (6) will be modified as shown in Figure 19. Thus, at step S28, the evolving processor (6) selects an individual for evolution in accordance with its fitness. However, before carrying out any evolutionary process, the evolving processor (6) checks the count in any invocation counter present in any structural element of that individual at step S29 and then modifies that individual in accordance with a count at step S30. For example, if the evolving processor (6) determines that the invocation count for a particular

gene exceeds a preset maximum, then the evolving processor (6) may determine that that particular gene may contain an infinite loop or be consuming excessive computational time and may eliminate that gene from the corresponding chromosome or may penalise the chromosome carrying that particular gene by reducing its fitness in accordance with the invocation count. At the other extreme, if the evolving processor (6) determines at step S29 that the count for a particular gene or functional element is zero, then the evolving processor (6) may determine that that particular functional element or gene is not performing any function and may cause that particular functional element or genes to be deleted at step S30.

Deletion of functional elements or genes that have an invocation count of zero may, however, limit evolutionary possibilities for future generations because, although such functional elements or genes may not be contributing directly to the current functionality, they may nevertheless comprise a useful genetic resource for future generations and immediate removal might therefore be deleterious to the evolutionary process. Therefore, in addition to the invocation counter, each functional element or gene may contain a redundancy counter for example redundancy counter (21f) in Figure 4b which indicates the number of generations for which it has been non-functional, that is the number of fitness evaluations over which its invocation count has remained zero. Figure 20 shows a

flowchart for illustrating the processes carried out by the evolving processor (6) in this circumstance. Thus, at step S31, the evolving processor (6) checks the invocation count for the structural element under consideration to determine whether the count is zero. If the answer is NO, then the evolving processor (6) proceeds to step S32 equivalent to step S30 in Figure 19 and may modify the fitness of the individual containing that structural element in accordance with the count, as described above, to penalise individuals where specific genes have high invocation count. In extreme cases, where the invocation count exceeds a preset maximum, then the evolving processor (6) may delete that structural element from the individual. Where the answer at step S31 is YES, the count is zero, then the evolving processor (6) takes one of two possible routes in accordance with respective probabilities PROB A andPROB B (steps S33 and S35). Where S33 is taken, then the evolving processor (6) will delete the structural element whose invocation count is zero. However, where step S35 is chosen, then at step S36, the evolving processor (6) will add one to the redundancy counter associated with that structural element and then check at step S37 whether the redundancy count has reached a preset limit x. If the answer at step S37 is NO, then the evolving processor will retain that particular structural element at step S39. If, however, the answer at step S37 is YES, then the evolving processor (6) will cause that particular structural element to be deleted at step S38.

It will, of course, be appreciated that the probabilities PROB A and PROB B may be zero or one so that either a structural element having an invocation count is always deleted or is always retained until the redundancy counter has reached a preset limit.

The computational penalty and redundancy counter mechanisms described above serve to reduce a recognised problem in evolutionary computing, that is the problem of bloat whereby functionally inactive parts of a gene, termed introns, are retained as evolutionary hangovers. Avoiding or reducing bloat avoids excessive complexity in the resulting computer program or mechanism for obtaining the solution so facilitating interpretation of the result of the evolutionary process.

Complexity may also be regulated by other mechanisms in additional to the computational penalty mechanism as described above. Thus, the complexity of a structural element may be constrained by penalising the fitness of the associated individual in accordance with its length (the number of structural elements forming that individual) or some other index such as, for example, the Kolmogorov complexity.

Clearly, as will be appreciated from the above, the fitness of an individual, a chromosome or even a gene within an individual may be determined at each fitness test and, for example, a gene may be deleted from an individual if it does not meet certain fitness criteria

In the arrangements described, structural elements may modify variables that are shared with other

structural elements so that, if two structural elements wish to modify the same shared variable, the final outcome may depend upon the order in which that modification occurs. In order to avoid this problem, chromosomes and the genes they carry may be executed sequentially or an execution order may be maintained for the genes.

Because, as described above, structural elements, especially genes, may share information, changes in one structural element may influence the fitness of other individuals within the populations. One way of accommodating this is to provide a so-called "steady state" strategy wherein one individual (or more where the evolutionary process is a crossover process) is selected from the population and a single evolutionary change is performed to a structural element of that individual and the fitness of that individual is re-evaluated in relation to the remainder of the population using a conventional fitness selection procedure such as described in the above mentioned book by Koza.

The above described examples assume that the user of the apparatus has little or no prior knowledge or experience of solving a particular type of problem by evolutionary processes. If, however, the individual does have such knowledge, then the individual may design the overall composition or structure one or more individuals within the initial population and may set specific evolutionary rates for those particular individuals so that, for example, they are retained throughout the

entirety of the evolutionary procedure, that is from one generation to the next or evolve at a rate much slower than other individuals. Similarly, a user may encode prior knowledge by setting values of certain parameters such as global variables.

The apparatus may also allow a user to select from or start from previous populations, sub-populations, individuals, chromosomes and genes as well as from previous or existing functional elements when initialising a new population.

The above described examples assume that a single population is provided. However, a complete population may be divided into a number of sub-populations, so-called demes and the evolutionary process carried out separately for each deme for a predetermined number of generations. This procedure is illustrated by the flow charts shown in Figure 21. Thus, at step S50, when the controller (2) defines the initial population of individuals, it separates that initial population randomly into two or more demes. At step S51, the controller causes the apparatus to carry out the evolutionary process shown in Figure 3 separately and independently for each deme. The controller (2) checks at step S52 the number of generations to which the evolutionary process has proceeded and if a predetermined number has not been reached, increments a counter x at step S53 and then repeats step S51 and S52 until the answer at step S52 is YES, the predetermined number of generations has been reached. At this stage, the

controller (2) determines at step S54 whether a maximum number of generations have been reached and if so terminates the process. If, however, the answer at step S54 is NO, then the controller (2) causes the theme populations to be modified at step S55 and repeats steps S51 to S54 until the answer at step S54 is YES.

Figure 22 shows in greater detail the steps that may be carried out at step S55. Thus, at step S56, the controller (2) selects at least two demes. When there are more than two demes, then the particular two demes may be selected at random. Then, at step S57, the controller (2) selects an individual or individuals from at least one of the two demes. This selection may be made in accordance with fitness and/or on a random basis. Then, at step S58, the controller (2) causes the selected individual or individuals to be transferred to the other of the two selected demes to produce modified deme populations. Steps S51 to S55 are then repeated until the answer at step S54 is YES. Thus, in this case, the deme populations are evolved separately but, after a predetermined number of generations, an individual or individuals may be transferred from one deme to another, so modifying the deme populations by allowing genetic material from the gene pool of one Deme to transfer to the gene pool for another deme.

An evolutionary process embodying the present invention is particularly well adapted for execution on a massively parallel computer system such as a Beowulf class cluster or a massively parallel multi-processor

supercomputer. This is especially the case where the total population is separate into demes because the evolutionary processing of each deme may be handled by a separate processor in the parallel system and the processors need only communicate with one another after the set predetermined number of generations. The evolutionary process may also be applied to a massively parallel system where a single individual population is being evolved. In this case, the current population store, variable store, process modifier store and data store may reside in an area of shared memory and individuals may be passed between processors in a cluster as simple data structures. In addition, the current population store (5) and gene store or pool (5a) may periodically be stored to non-volatile storage to enable ready resumption of an evolutionary process if, for example, the computer system on which the processor is operating fails or undergoes a power interruption. This is of particular benefit in a multi-node cluster system where there is an increased likelihood of failure of at least one of the nodes in a given amount of time compared to a single processor system.

As will be appreciated from the above, unlike the genetic programming approach proposed by Koza, individuals, chromosomes, genes and functional elements provided by apparatus embodying the present invention may adopt many different topologies, dependent upon the problem to be solved. Figure 23 illustrates some of the possibilities. Thus, Figure 23a shows the linear

representation described above while Figure 26b shows a representation analogous to the tree structure of Koza, Figure 23c shows a neural network type representation and Figure 23b shows a Bayesian belief network representation.

In the examples described above, the rules forming a chromosome are activated in the order determined by those rules, that is by the content of the chromosome. However, a random evolutionary "start" operator may be used.

In the examples described above, a fitness value is associated with an individual because the fitness that matters is that of the individual. Thus, the capability of any individual is there only as a result of the combination of the various elements therein, so that, while the combination might have high fitness, it is not necessarily true that any one of its constituents is, singly, especially fit. However, it may well be advantageous to make available to future populations the constituents (structural elements) of particularly fit individuals. This could be achieved by assessing the fitness of a structural element in terms of the number of times, over multiple populations, that particular chromosomes, genes, or functional elements appear in individuals with high fitness. Another way would be to make different levels within an individual (for example the chromosomes or genes) "mini-individuals" and to allow them to co-evolve in which case each mini-individual will have its own fitness value.

An embodiment of the present invention provides a procedure for simultaneously deriving multiple mechanisms, programs or computer functions for solving complex problems following a procedure analogous to natural evolution wherein individuals in a population have a structure closely analogous to the structure of a biological genome. This enables naturally occurring evolutionary functions to be emulated including duplication, mutation, crossover, transposition, linkage, cross regulation, operons, polyploid, dominance and recession, deletion, insertion and transfer of information between individuals.

Part of the evolutionary procedure described above has been used to analyse data from DNA micro-array hybridisation experiments wherein each data point represents the ratio of the (transcriptional) expression levels of a particular gene under two or more different experimental conditions. The results of this evaluation are described in a paper entitled "Genomic Computing: Explanatory Modelling for Functional Genomics" by the inventors to be published in the Real World Applications section of the Proceedings of the Genetic and Evolutionary Computing Conference, July 2000 (GECCO 2000), the whole contents of which are incorporated herein by reference.

**CLAIMS:**

1. Apparatus for enabling a mechanism or program for providing a solution to a problem to be evolved from an initial population of individuals each of which represents a potential mechanism or program for providing a solution to the problem, the apparatus comprising:

generating means operable to provide an initial population of individuals each having a plurality of structural elements each comprising a plurality of functional components defining operations to be performed on at least one input to the functional component;

activating means for activating each individual to cause the operations defined by the functional components of that individual to be performed to produce a result;

evaluating means for determining from each result a fitness of the corresponding individual for providing a mechanism or program for providing a solution to the problem;

evolving means operable to produce a new generation of individuals by causing an evolutionary process to occur involving at least one structural element and at least one functional component; and

controlling means for controlling the activating, evaluating and evolving means so as to cause each structural element of the new generation to be activated, its fitness to be determined and an evolutionary process to occur in at least one structural element and at least one of functional component to produce another generation

of individuals for a succession of generations until either the evaluating means determines that an individual of the current generation represents a best obtainable or suitable mechanism or program for providing a solution to the problem or a predetermined number of generations have been evolved.

2. Apparatus according to claim 1, wherein the generating means is operable to provide each individual as a hierarchical structure having at least one first structural element comprising a plurality of second structural elements each comprising at least one functional component.

3. Apparatus according to claim 1, wherein generating means is operable to provide each structural element such that the structural element emulates a gene.

4. Apparatus according to claim 1, wherein the generating means is operable to provide each individual as a hierarchical structure having at least one first structural element emulating a chromosome and comprising a plurality of second structural elements each emulating a gene, with each second structural element comprising at least one functional component.

5. Apparatus according to claim 1, wherein the generating means is operable to provide each individual as a hierarchical structure having a plurality of first

structural elements each emulating a chromosome and each comprising at least one second structural element emulating a gene with each second structural element comprising at least one functional component.

6. Apparatus according to any one of the preceding claims, wherein the evolving means is operable to produce a new generation of individuals by causing an evolutionary process to occur involving any structure element and any functional component.

7. Apparatus according to any one of claims 2 to 5, wherein the evolving means is operable to cause respective separate evolutionary processes to occur involving at least one first structural element, at least one second structural element and at least one functional component.

8. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide each functional component as a descriptor defining an operation to be performed by the functional component and one or more arguments representing values usable in the defined operation.

9. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide each structural element as a descriptor identifying the structural component and data identifying

the one or more structural elements or functional components forming that structural element.

10. Apparatus according to claim 4 or 5, wherein the generating means is operable to provide each structural element such that each second structural element forms a functional component of the corresponding first structural element and the generating means is operable to provide each structural element as a descriptor identifying the structural component and the number of functional components forming the structural element and data identifying the one or more functional components forming that structural element.

11. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide functional components each adapted to carry out an operation selected from the following:

a numeric operation;

a logic operation;

a computer program operation.

12. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide at least one functional component adapted to carry out a numeric operation selected from the following:

add, subtract, divide, multiply.

13. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide at least one functional component adapted to carry out a Boolean logic operation.

14. Apparatus according to any one of the preceding claims, wherein the evolving means is operable to carry out an evolving process by causing a mutation in a structural element or functional component.

15. Apparatus according to any one of the preceding claims, wherein the evolving means is operable to carry out an evolving process by causing duplication of a structural element or functional component.

16. Apparatus according to any one of the preceding claims, wherein the evolving means is operable to carry out an evolving process by causing transposition of functional components in a structural element.

17. Apparatus according to any one of the preceding claims, wherein the evolving means is operable to carry out an evolving process by causing crossover of functional components between structural elements.

18. Apparatus according to any one of the preceding claims, wherein the evolving means is operable to carry out an evolving process by causing crossover of

functional components or structural elements between individuals.

19. Apparatus according to claim 3, 4 or 5, wherein the evolving means is operable to carry out at least one of the following evolving processes:

duplication, mutation, crossover, transposition, deletion and insertion.

20. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide a set of global variables accessible by all individuals.

21. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide at least one structural element with a set of local variables specific to that structural element or individual.

22. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide at least one evolutionary process modifying parameter and to provide the individuals such that at least one structural element or one functional element is affected by said at least one or a evolutionary process modifying parameter.

23. Apparatus according to claim 3, 4 or 5, wherein the generating means is operable to provide at least a evolutionary process modifying parameter which mimics a metabolite that affects the operation of at least one gene.

24. Apparatus according to any one of the preceding claims, wherein the generating means is operable to provide for communication between structural elements such that an output from one structural element affects another.

25. Apparatus according to claim 3, 4, 5, 19 or 23, wherein the generating means is operable to provide for communication between genes such that an output from one gene affects another.

26. Apparatus according to any one of the preceding claims, wherein the evaluating means comprises means for weighting the fitness of an individual on the basis of the complexity of that individual.

27. Apparatus according to any one of the preceding claims, comprising counting means for determining the number of times a functional component is activated.

28. Apparatus according to claim 27, wherein the evaluating means comprises means for weighting the

fitness of an individual on the basis of the output of said counting means.

29. Apparatus according to claim 27 or 28, wherein the evaluating means is operable to delete a functional component or structural element if the count exceeds a predetermined number.

30. Apparatus according to claim 27, 28 or 29, wherein the evaluating means is operable to delete a functional component or structural element if the count is zero.

31. Apparatus according to claim 27, 28 or 29, wherein the evaluating means is operable to delete a functional component or structural element if the count is zero for a predetermined number of generations.

32. Apparatus according to any one of the preceding claims, comprising validation means for testing a mechanism or program represented by an individual against data not previously available for that individual.

33. A massively parallel computer apparatus programmed to provide apparatus in accordance with any one of the preceding claims.

34. In a computer system, a method of enabling a mechanism or program for providing a solution to a problem to be evolved from an initial population of

individuals each of which represents a potential mechanism or program for providing a solution to the problem, the method comprising the steps of:

providing an initial population of individuals each having a plurality of structural elements each comprising a plurality of functional components defining operations to be performed on at least one input to the functional component;

activating each individual to cause the operations defined by the functional components of that individual to be performed to produce a result;

determining from each result a fitness of the corresponding individual for providing a mechanism or program for providing a solution to the problem;

producing a new generation of individuals by causing an evolutionary process to occur involving at least one structural element and at least one functional component; and

repeating the activating, evaluating and evolving steps for a succession of generations until either an individual of the current generation represents a best obtainable or suitable mechanism or program for providing a solution to the problem or a predetermined number of generations have been evolved.

35. A method according to claim 34, which comprises providing each individual as a hierarchical structure having at least one first structural element comprising

a plurality of second structural elements each comprising at least one functional component.

36. A method according to claim 34, which comprises providing each structural element such that the structural element emulates a gene.

37. A method according to claim 34, which comprises providing each individual as a hierarchical structure having at least one first structural element emulating a chromosome and comprising a plurality of second structural elements each emulating a gene, with each second structural element comprising at least one functional component.

38. A method according to claim 34, which comprises providing each individual as a hierarchical structure having a plurality of first structural elements each emulating a chromosome and each comprising at least one second structural element emulating a gene with each second structural element comprising at least one functional component.

39. A method according to any one of claims 34 to 38, which comprises evolving a new generation of individuals by causing an evolutionary process to occur involving any structure element and any functional component.

40. A method according to any one of claims 35 to 38, which comprises causing respective separate evolutionary processes to occur involving at least one first structural element, at least one second structural element and at least one functional element.

41. A method according to any one of claims 34 to 40, which comprises providing each functional component as a descriptor defining an operation to be performed by the functional component and one or more arguments representing values usable in the defined operation.

42. A method according to any one of claims 34 to 41, which comprises providing each structural element as a descriptor identifying the structural component and data identifying the one or more structural elements or functional components forming that structural element.

43. A method according to claim 37 or 38, which comprises providing each structural element such that each second structural element forms a functional component of the corresponding first structural element and the generating means is operable to provide each structural element as a descriptor identifying the structural component and the number of functional components forming the structural element and data identifying the one or more functional components forming that structural element.

44. A method according to any one of claims 34 to 43, which comprises providing functional components each adapted to carry out an operation selected from the following:

a numeric operation;

a logic operation;

a computer program operation.

45. A method according to any one of claims 34 to 44, which comprises providing at least one functional component adapted to carry out a numeric operation selected from the following:

add, subtract, divide, multiply.

46. A method according to any one of claims 34 to 45 which comprises providing at least one functional component adapted to carry out a Boolean logic operation.

47. A method according to any one of claims 34 to 46, which comprises carrying out an evolving process by causing a mutation in a structural element or functional component.

48. A method according to any one of claims 34 to 47, which comprises carrying out an evolving process by causing duplication of a structural element or functional component.

49. A method according to any one of claims 34 to 48, which comprises carrying out an evolving process by causing transposition of functional components in a structural element.

50. A method according to any one of claims 34 to 49, which comprises carrying out an evolving process by causing crossover of functional components between structural elements.

51. A method according to any one of claims 34 to 50, which comprises carrying out an evolving process by causing crossover of functional components or structural elements between individuals.

52. A method according to claim 36, 37 or 38, which comprises carrying out at least one of the following evolving processes:

duplication, mutation, crossover, transposition, deletion and insertion.

53. A method according to any one of claims 34 to 52, which comprises providing a set of global variables accessible by all individuals.

54. A method according to any one of claims 34 to 53, which comprises providing at least one structural element with a set of local variables specific to that structural element or individual.

55. A method according to any one of claims 34 to 54, which comprises providing at least one evolutionary process modifying parameter and providing the individuals such that at least one structural element or one functional element is affected by said at least one or a evolutionary process modifying parameter.

56. A method according to claim 36, 37, 38 or 52, which comprises providing at least an evolutionary process modifying parameter which mimics a metabolite that affects the operation of at least one gene.

57. A method according to any one of claims 34 to 56, which comprises providing for communication between structural elements such that an output from one structural element affects another.

58. A method according to claim 36, 37, 38, 52 or 56, which comprises providing for communication between genes such that an output from one gene affects another.

59. A method according to any one of claims 34 to 57, which comprises weighting the fitness of an individual on the basis of the complexity of that individual.

60. A method according to any one of claims 34 to 59, which comprises determining the number of times a functional component is activated.

61. A method according to claim 60, which comprises weighting the fitness of an individual on the basis of the number of times a functional component is activated.

62. A method according to claim 60 or 61, which comprises deleting a functional component or structural element if a predetermined number is exceeded.

63. A method according to claim 60, 61 or 62, which comprises deleting a functional component or structural element if the number is zero.

64. A method according to claim 60, 61 or 62, which comprises deleting a functional component or structural element if the number is zero for a predetermined number of generations.

65. A method according to any one of claims 34 to 64, comprising validating by testing a mechanism or program represented by an individual against data not previously available for that individual.

66. Apparatus for enabling a structure, mechanism or program for providing a solution to a problem to be evolved from an initial population of individuals each of which represents a potential way, structure, mechanism or program for providing a solution to the problem, the apparatus comprising a processor arrangement operable to provide an initial population of individuals each having

a plurality of structural elements each comprising at least one functional component defining operations to be performed on at least one input to the functional component and then to

(i) activate each individual to cause the operation defined by the at least one functional component of that individual to be performed to produce a result;

(ii) determine from each result a fitness of the corresponding individual for providing a mechanism or program for providing a solution to the problem;

(iii) produce a new generation of individuals by causing an evolutionary process to occur involving at least one structural element and at least one functional component; and then to repeat (i) to (iii) until either an individual of the current generation represents a suitable way, structure, mechanism or program for providing a solution to the problem or a predetermined number of generations have been evolved.

67. Apparatus for enabling a structure, mechanism or program for providing a solution to a problem to be evolved from an initial population of individuals each of which represents a potential way, structure, mechanism or program for providing a solution to the problem, the apparatus comprising a processor arrangement operable to provide an initial population of individuals each representing a genome consisting of one or more chromosomes each consisting of a plurality of genes each comprising at least one functional component defining an

operation to be performed on at least one input to the functional component and then to

(i) activate each individual to cause the operations defined by the functional components of that individual to be performed to produce a result;

(ii) determine from each result a fitness of the corresponding individual for providing a mechanism or program for providing a solution to the problem;

(iii) produce a new generation of individuals by causing an evolutionary process to occur involving at least one gene and at least one functional component; and then to repeat (i) to (iii) until either an individual of the current generation represents a suitable way, structure, mechanism or program for providing a solution to the problem or a predetermined number of generations have been evolved.

68. Apparatus according to claim 66 or 67, wherein the processor arrangement is operable to provide functional components each adapted to carry out an operation selected from the following:

a numeric operation;

a logic operation;

a computer program operation.

69. Apparatus according to claim 67, wherein the processor arrangement is operable to carry out at least one of the following evolving processes:

duplication, mutation, crossover, transposition, deletion and insertion, on a gene or plurality of genes.

70. Apparatus according to claim 67 or 69, wherein the processor arrangement is operable to provide at least one evolutionary process modifying parameter that mimics the effect of a metabolite on a gene.

71. In a computer apparatus having a processor arrangement, a method of providing a population of individuals each representing a potential way, structure, mechanism or program for providing a solution to a problem, in which method the population of individuals is generated such that each individual has a plurality of structural elements each comprising a plurality of functional components defining operations to be performed on at least one input to the functional component, wherein at least one structural element and at least one function element is capable of modification to enable evolution of the population.

72. In a computer apparatus having a processor arrangement, a method of providing a population of individuals each representing a potential way, structure, mechanism or program for providing a solution to a problem, in which method the population of individuals is generated such that each individual mimics a genome having at least one chromosome comprising a plurality of genes each comprising at least one functional element

defining a functional operation or operations to be carried out when the gene is activated, wherein at least one gene and at least one functional element is capable of modification to enable evolution of the population.

73. A signal carrying processor implementable instructions for causing a processor or processor arrangement to become configured to form an apparatus in accordance with any one of claims 1 to 33 or 66 to 70.

74. A storage medium carrying processor implementable instructions for causing a processor or processor arrangement to become configured to form an apparatus in accordance with any one of claims 1 to 33 or 66 to 70.

75. A signal carrying processor implementable instructions for causing a processor or processor arrangement to become configured to carry out a method in accordance with any one of claims 34 to 65, 71 or 72.

76. A signal carrying processor implementable instructions for causing a processor or processor arrangement to become configured to carry out a method in accordance with any one of claims 34 to 65, 71 or 72.

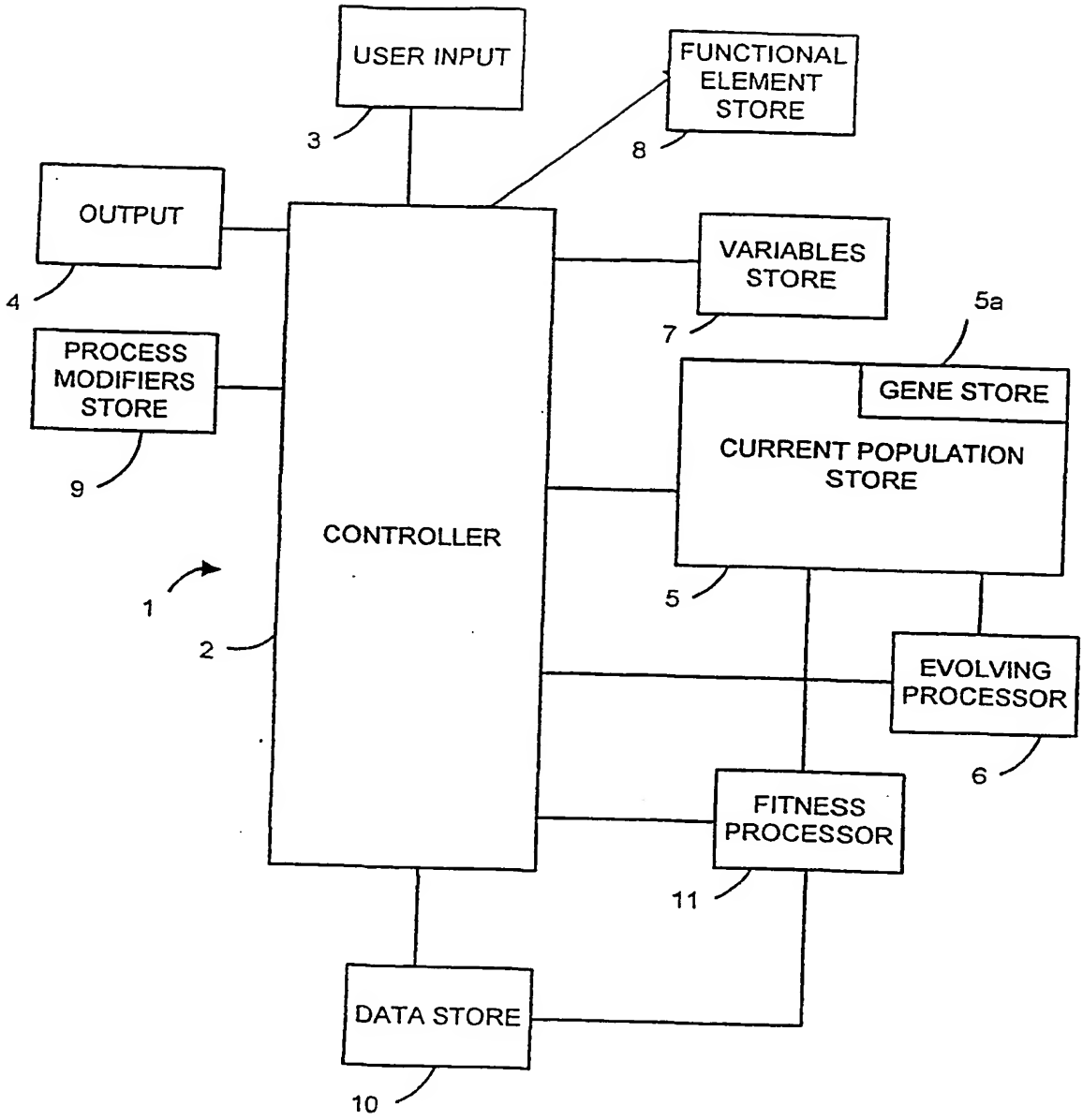



FIG.1

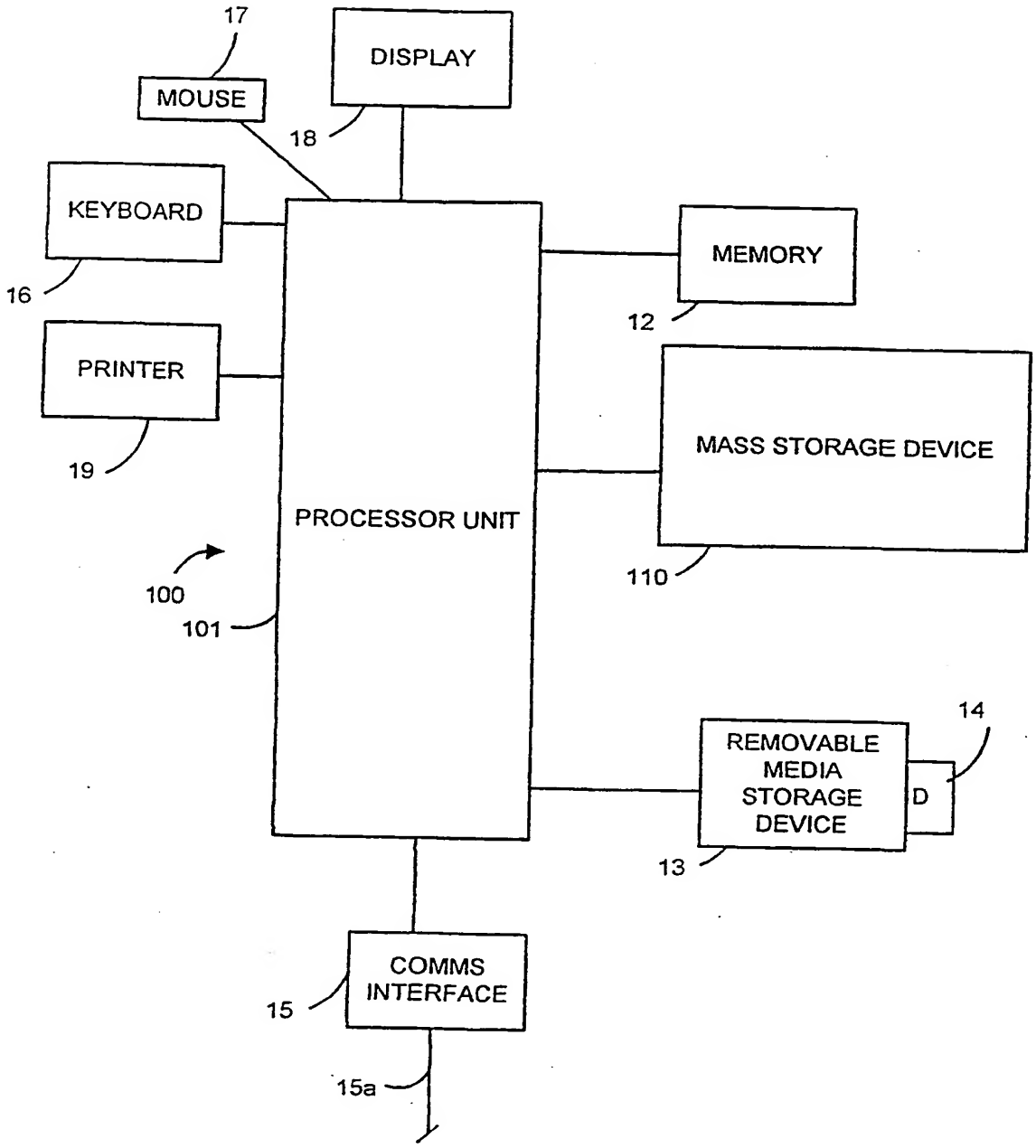

17
MOUSE
DISPLAY
18
KEYBOARD
16
MEMORY
12
PRINTER
19
MASS STORAGE DEVICE
110
PROCESSOR UNIT
100
101
14
REMOVABLE MEDIA STORAGE DEVICE
D
13
COMMS INTERFACE
15
15a

FIG.2

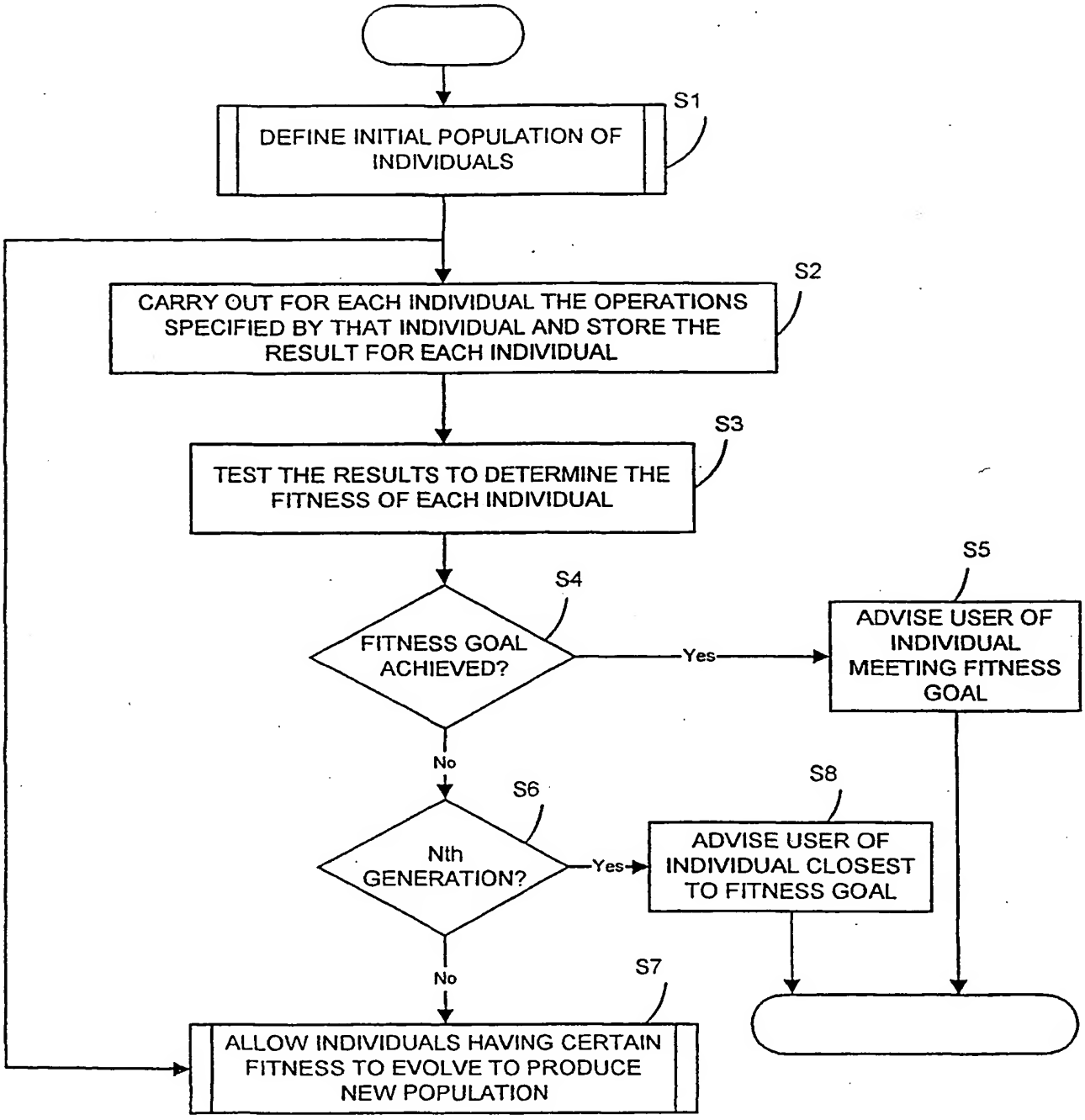



FIG.3

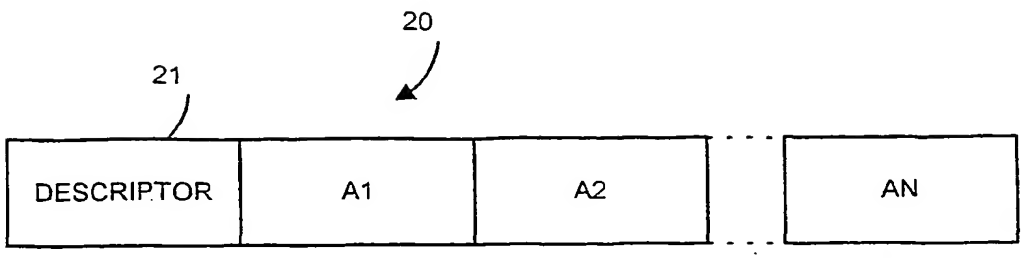



FIG. 4a

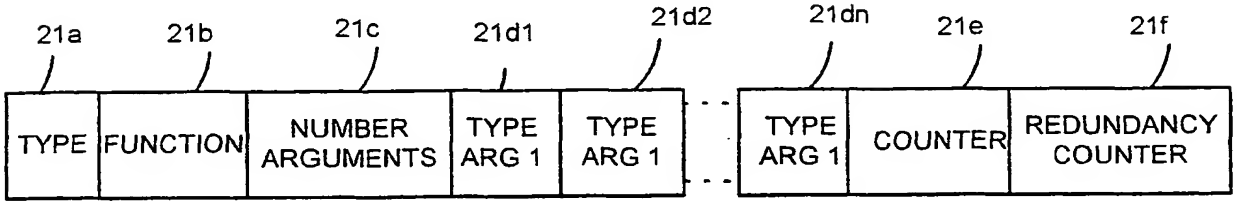



FIG. 4b

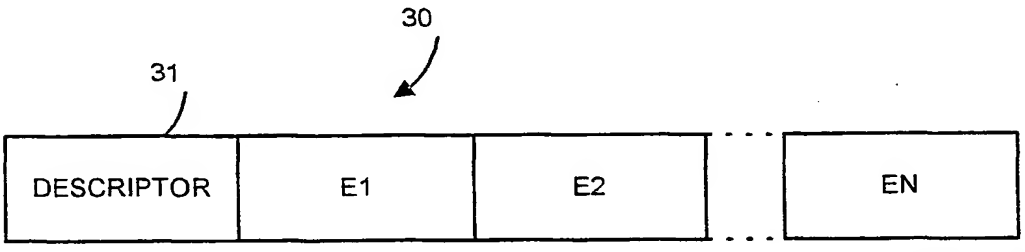



FIG. 4c

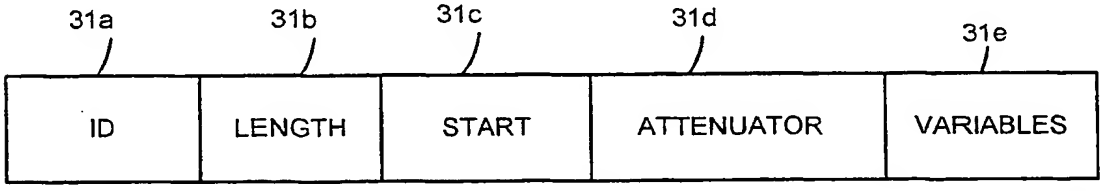



FIG. 4d

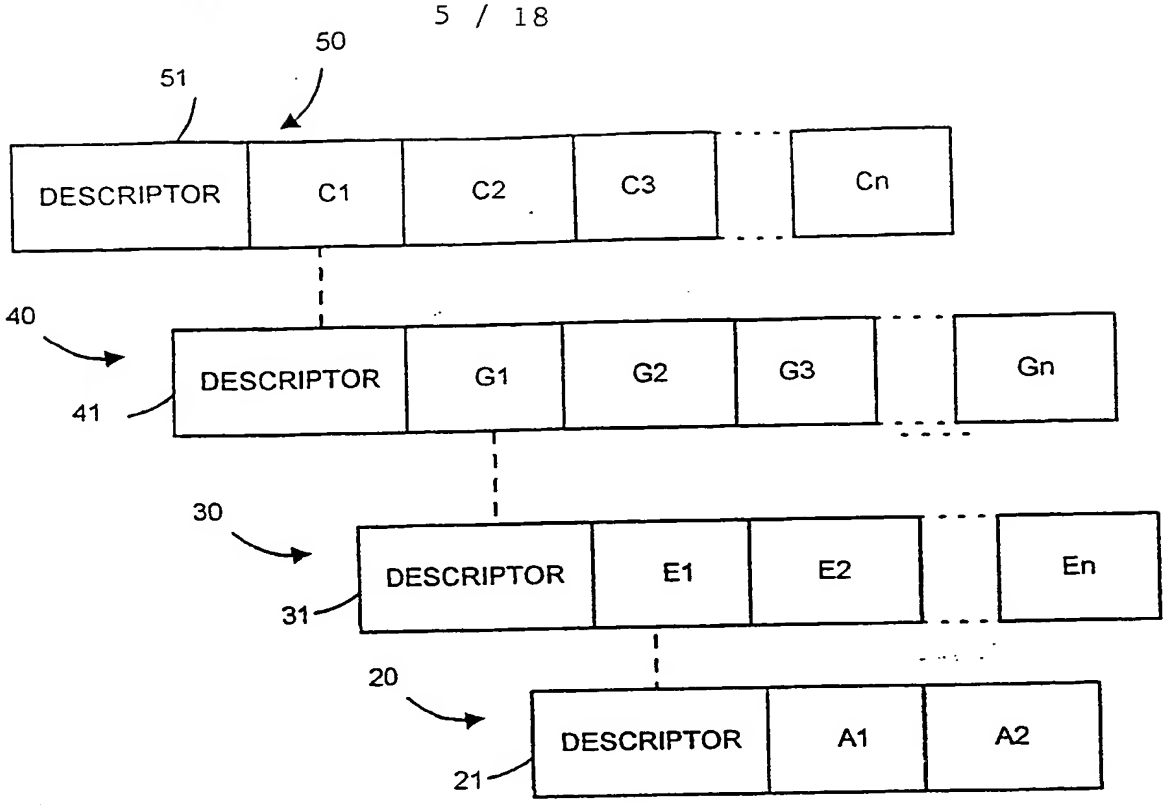



FIG. 5

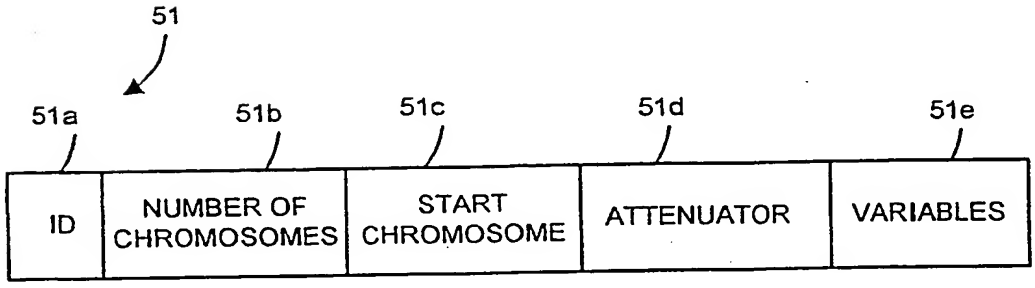



FIG. 6

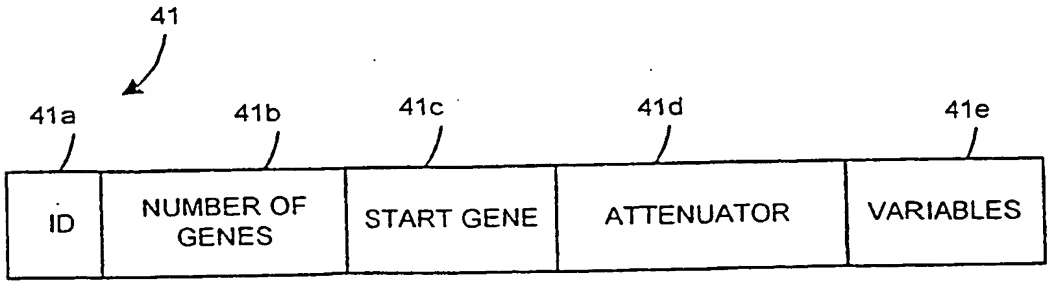
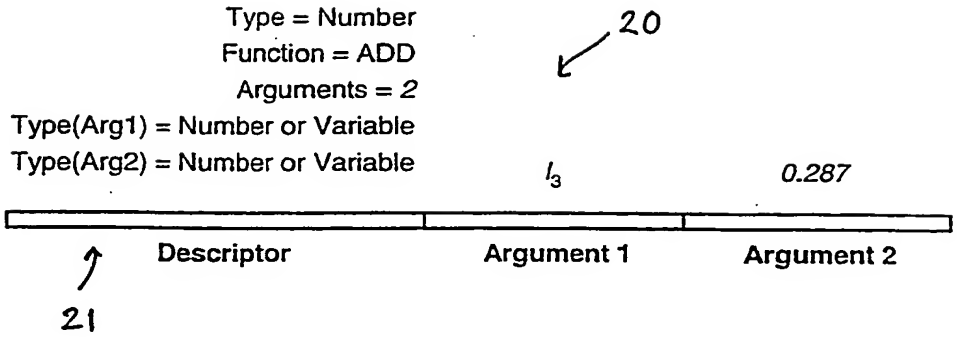
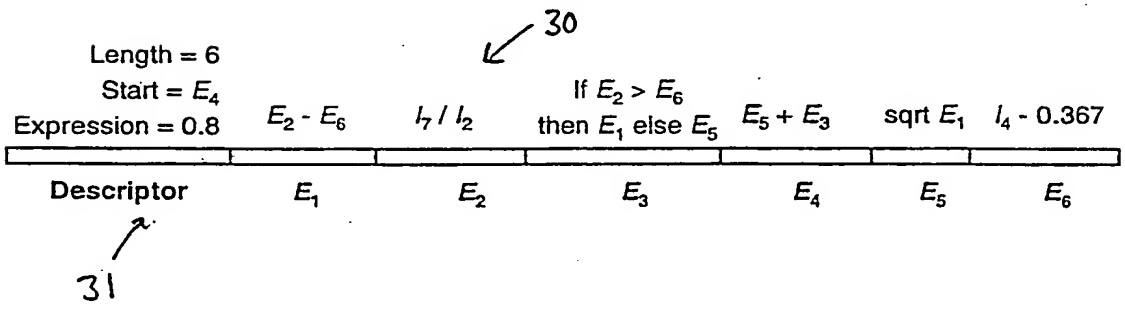



FIG. 7




$$output = I_3 + 0.287$$

FIG. 8




$$output = 0.8 \times \sqrt{\frac{I_7}{I_2} - I_4 + 0.367} + if\ \frac{I_7}{I_2} > (I_4 - 0.367)\ then \left( \frac{I_7}{I_2} - I_4 + 0.367 \right) else \sqrt{\frac{I_7}{I_2} - I_4 + 0.367}$$

FIG. 9

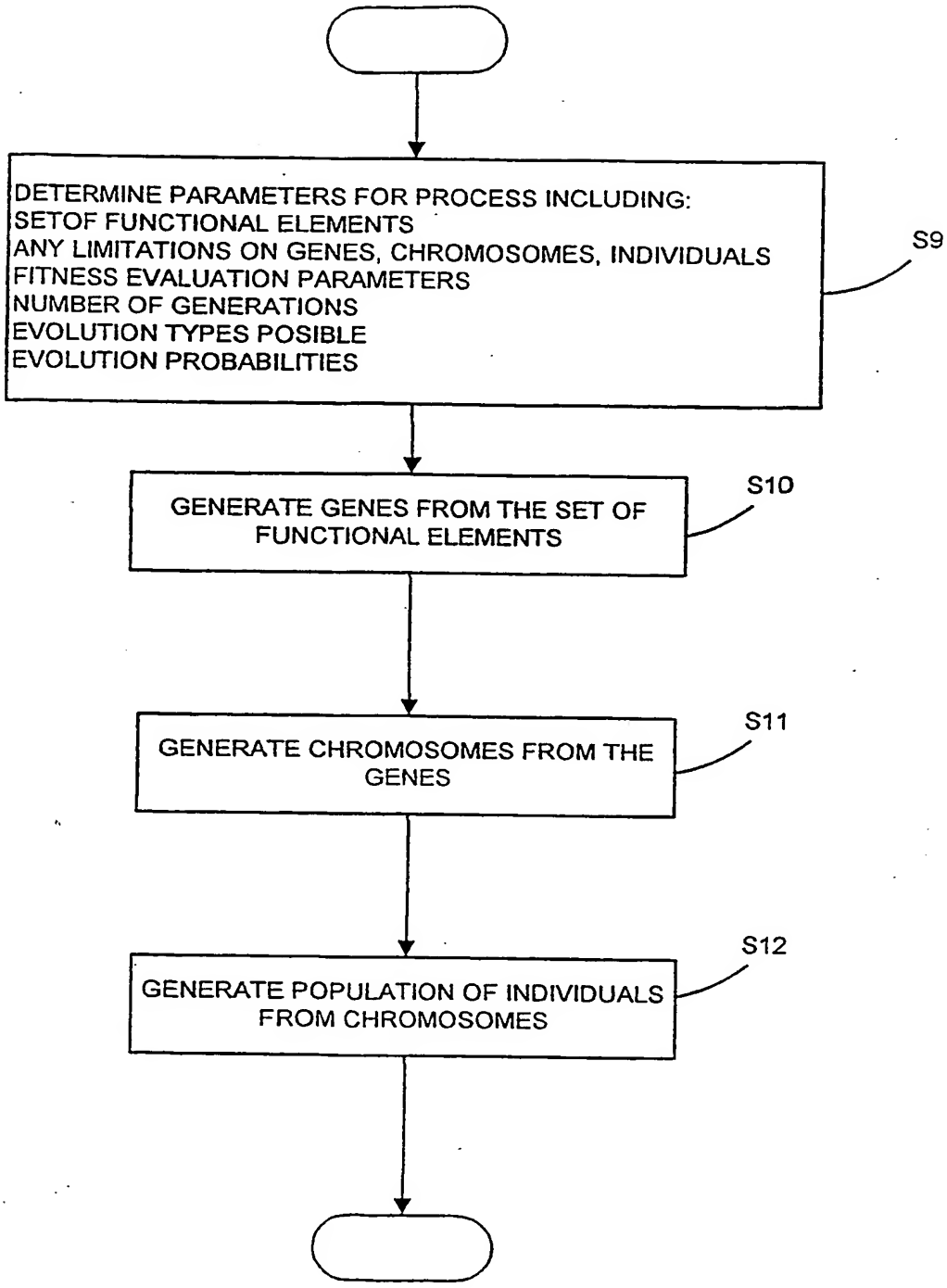

DETERMINE PARAMETERS FOR PROCESS INCLUDING:
SETOF FUNCTIONAL ELEMENTS
ANY LIMITATIONS ON GENES, CHROMOSOMES, INDIVIDUALS
FITNESS EVALUATION PARAMETERS
NUMBER OF GENERATIONS
EVOLUTION TYPES POSIBLE
EVOLUTION PROBABILITIES
S9
GENERATE GENES FROM THE SET OF FUNCTIONAL ELEMENTS
S10
GENERATE CHROMOSOMES FROM THE GENES
S11
GENERATE POPULATION OF INDIVIDUALS FROM CHROMOSOMES
S12

FIG. 10

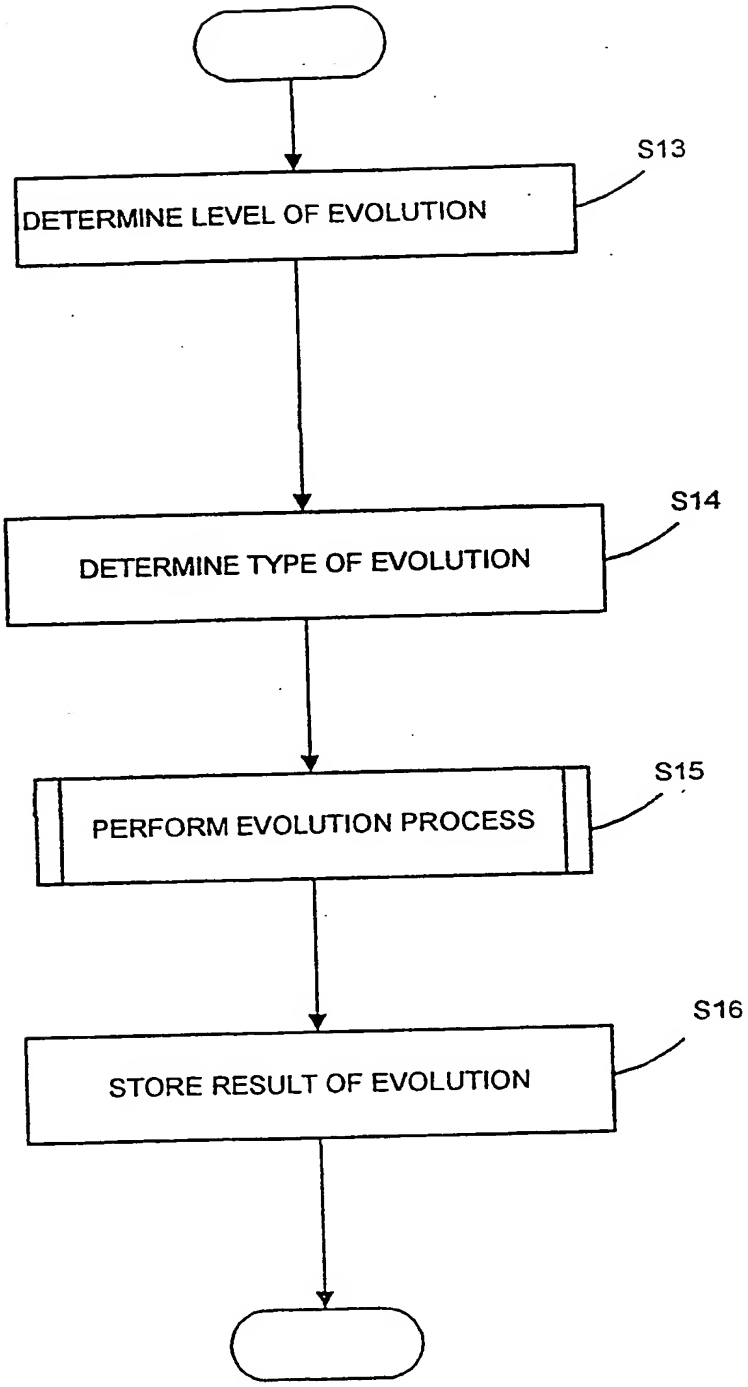

S13
DETERMINE LEVEL OF EVOLUTION
S14
DETERMINE TYPE OF EVOLUTION
S15
PERFORM EVOLUTION PROCESS
S16
STORE RESULT OF EVOLUTION

FIG. 11

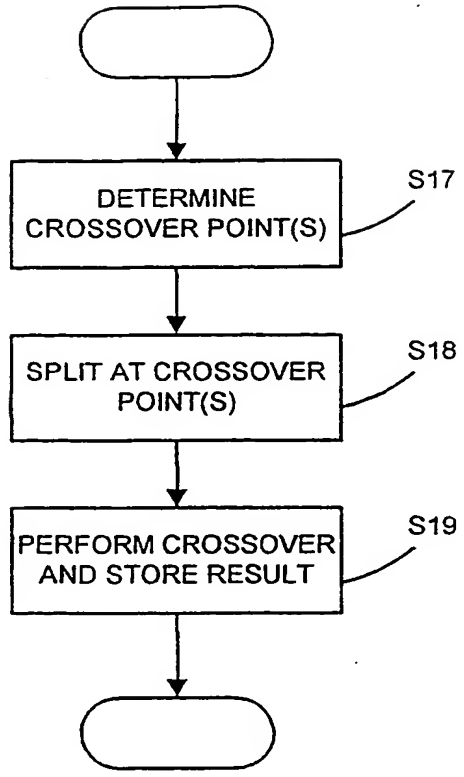



FIG. 12a

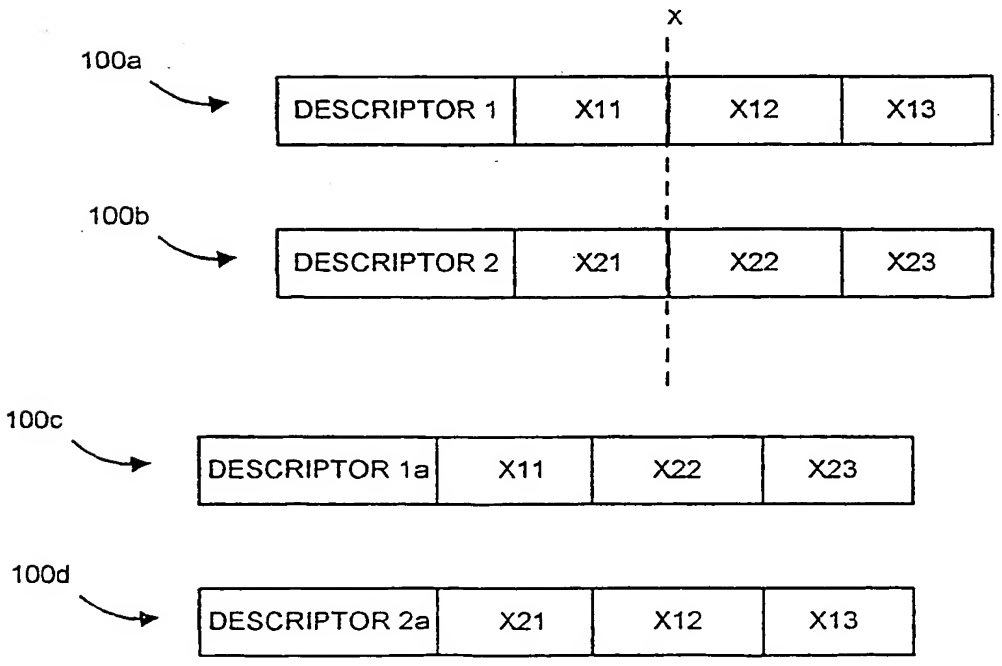



FIG. 12b

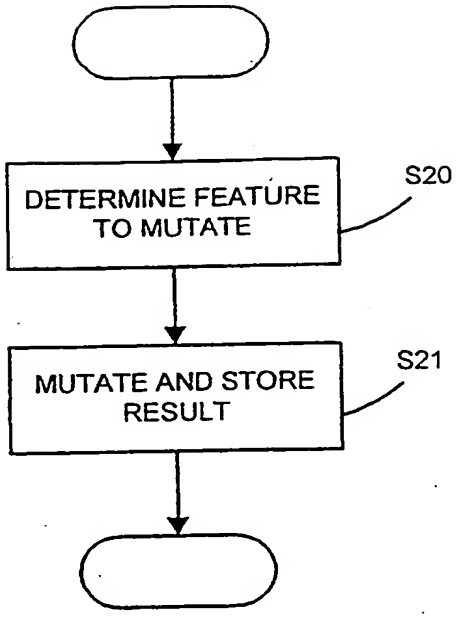



FIG. 13a

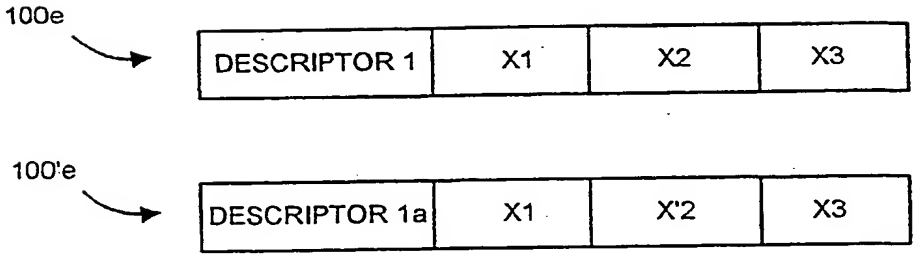



FIG. 13b

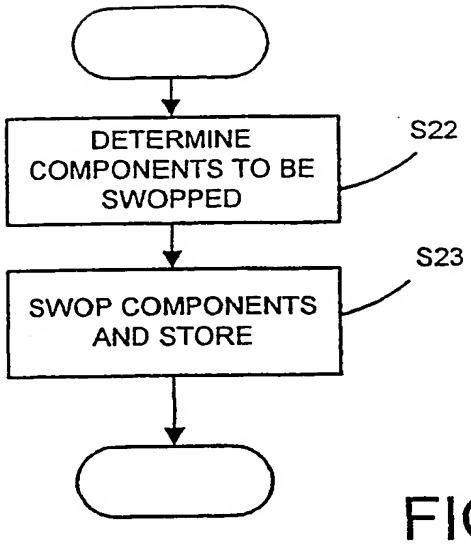



FIG. 14a

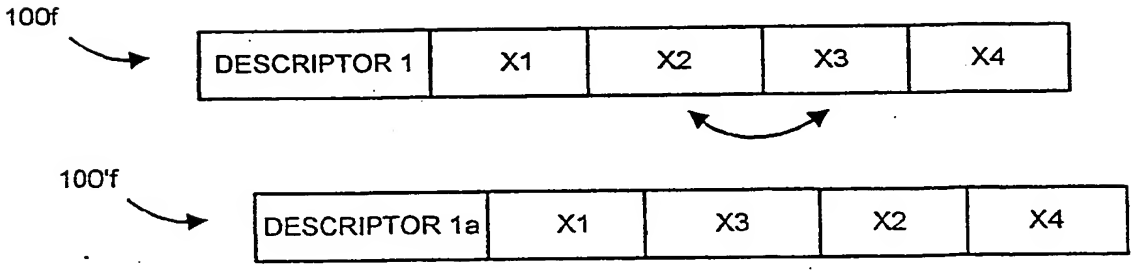



FIG. 14b

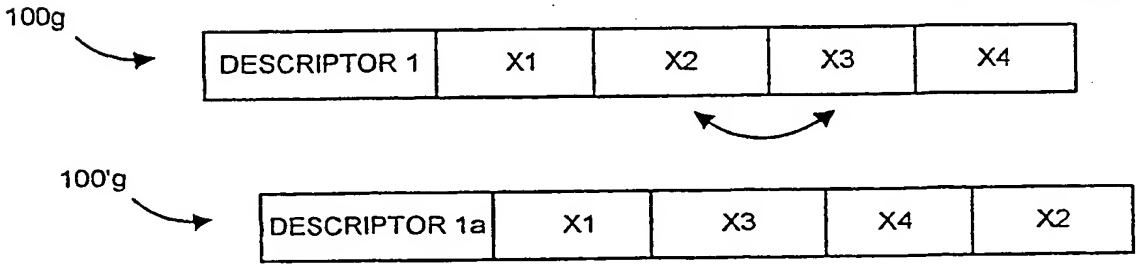



FIG. 14c

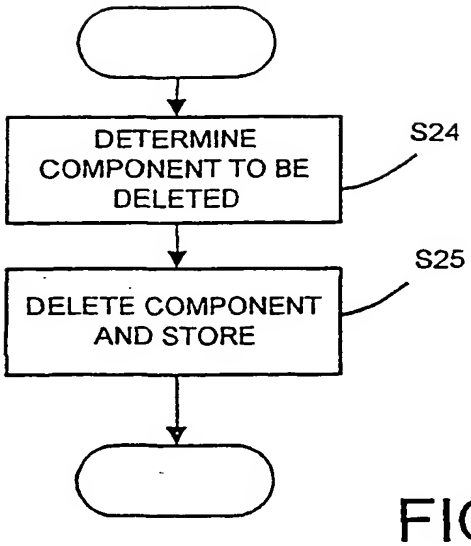



FIG. 15a

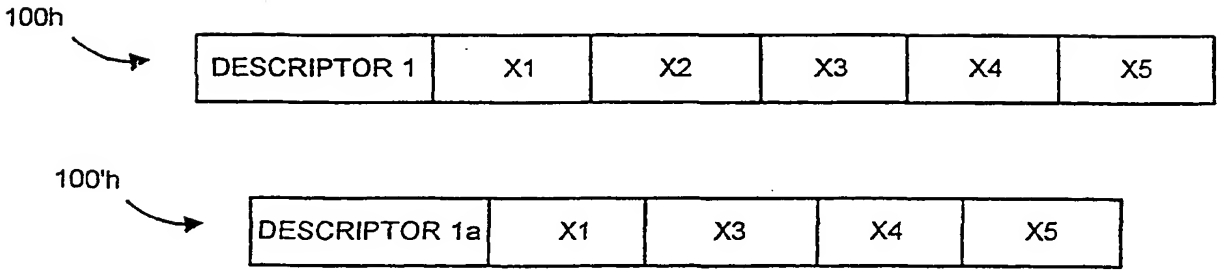



FIG. 15b

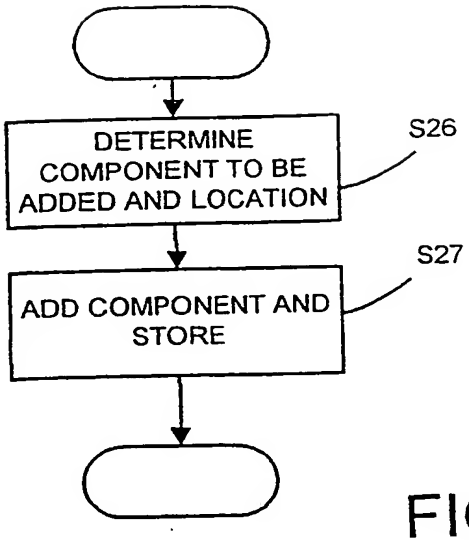



FIG. 16a

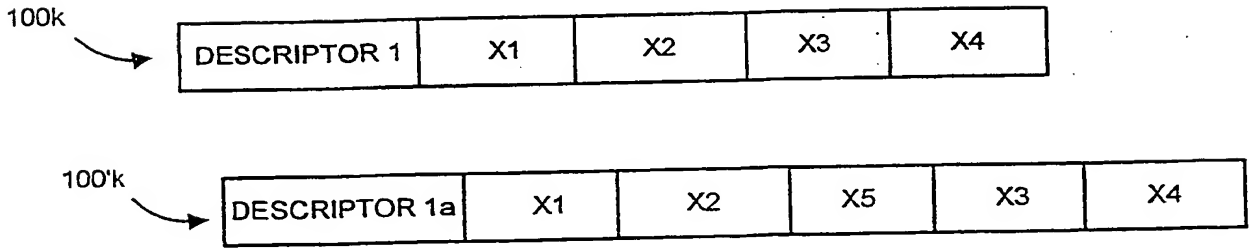



FIG. 16b

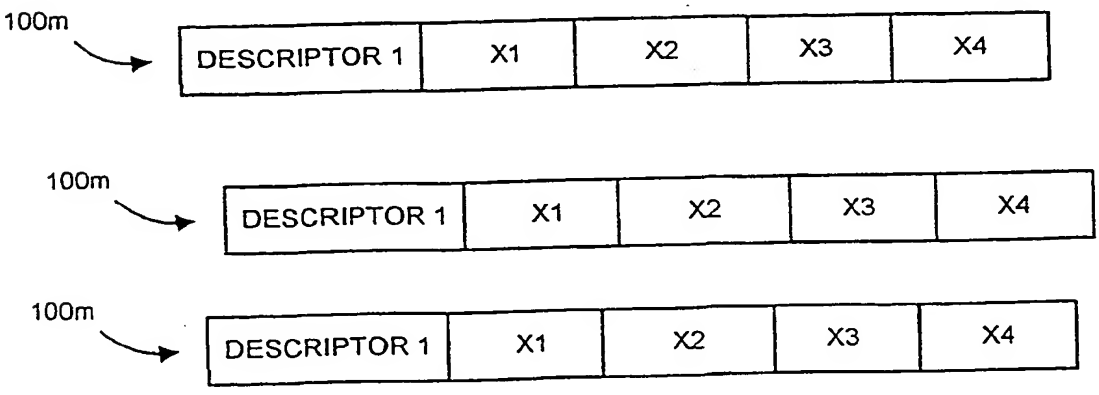



FIG. 17

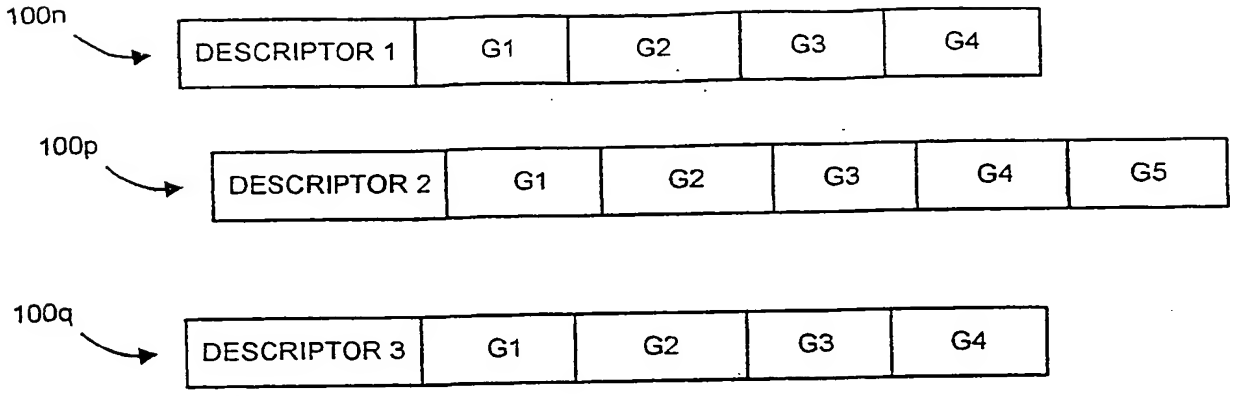



FIG. 18a

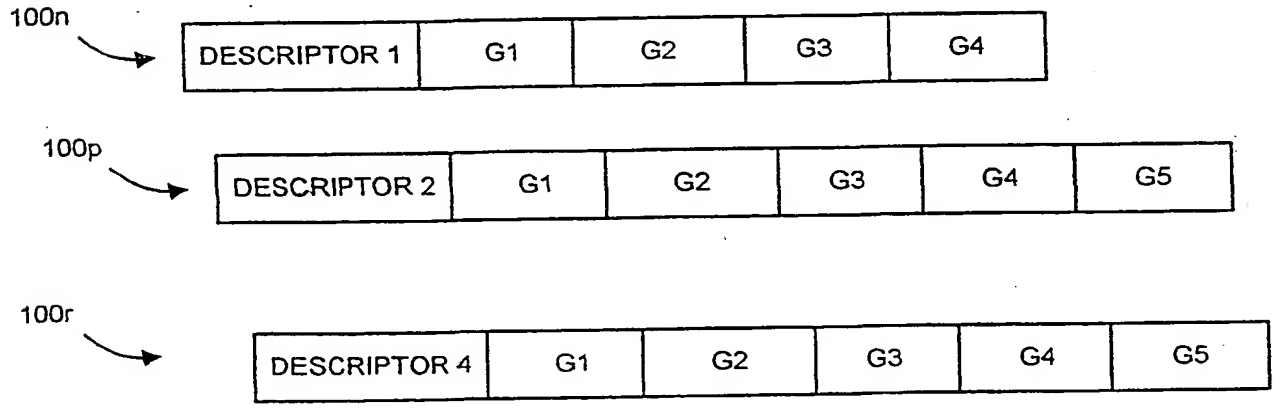



FIG. 18b

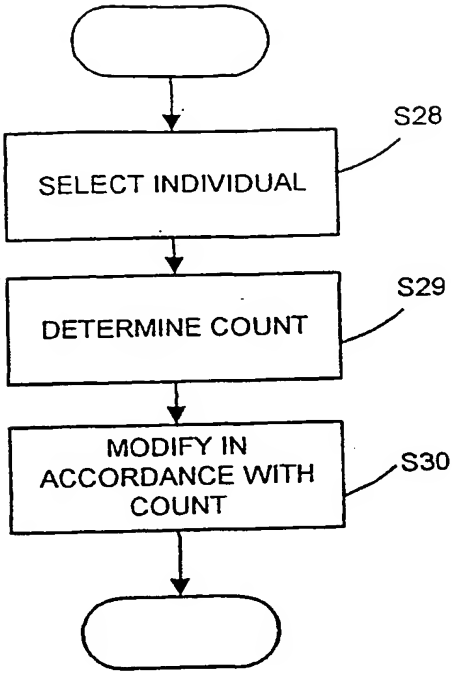



FIG. 19

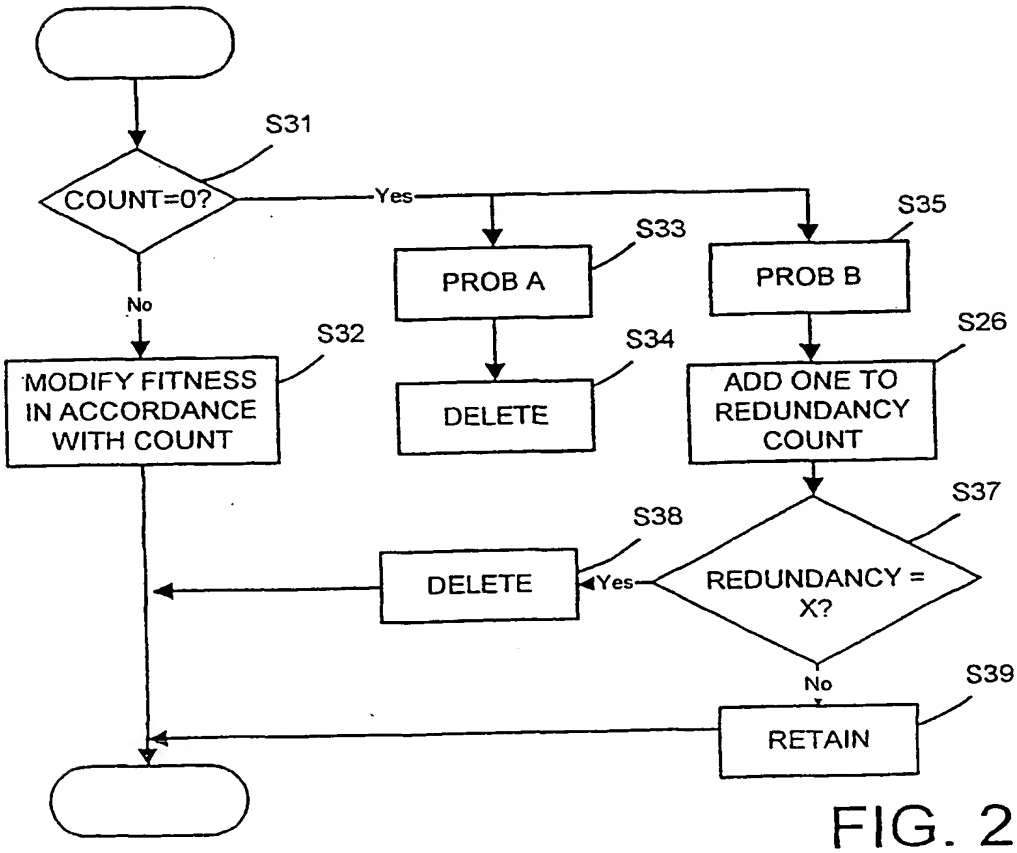



FIG. 20

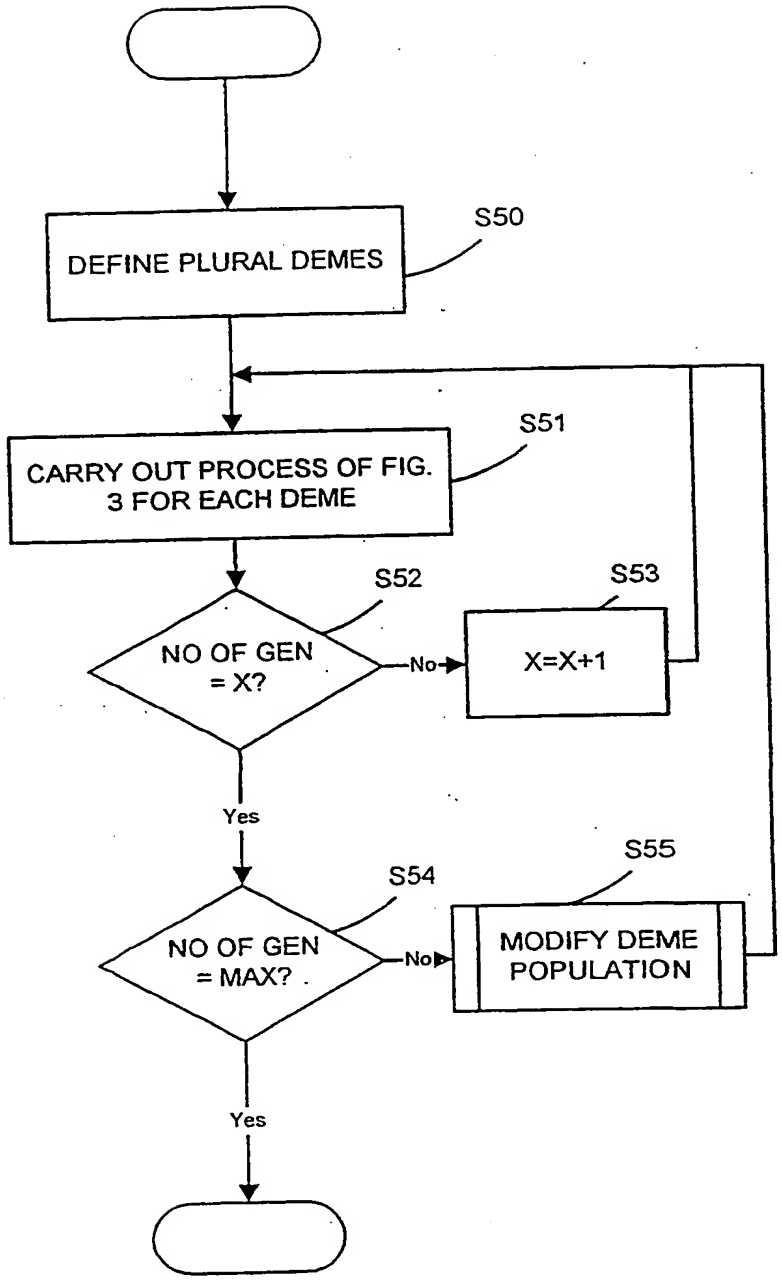

DEFINE PLURAL DEMES
S50
CARRY OUT PROCESS OF FIG. 3 FOR EACH DEME
S51
NO OF GEN = X?
S52
No
X=X+1
S53
Yes
NO OF GEN = MAX?
S54
No
MODIFY DEME POPULATION
S55
Yes

FIG. 21

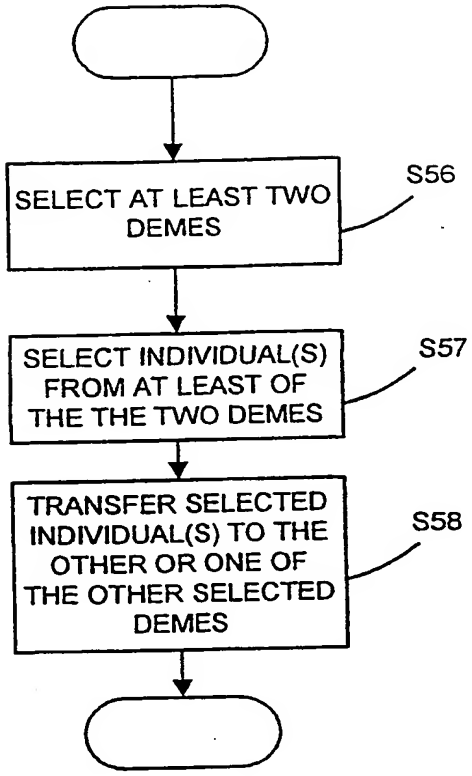



FIG. 22

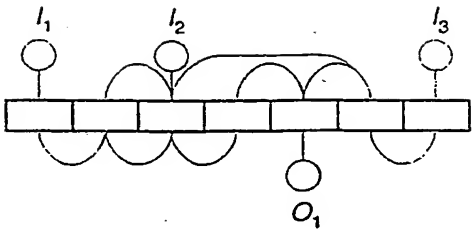



FIG. 23a

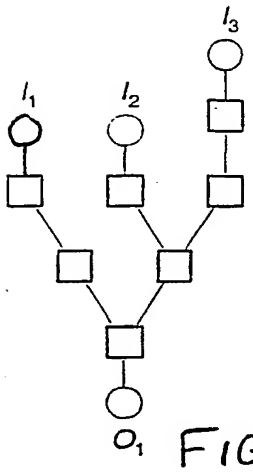



FIG. 23b

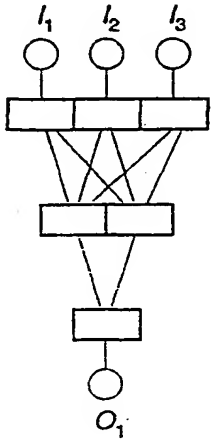



FIG. 23c

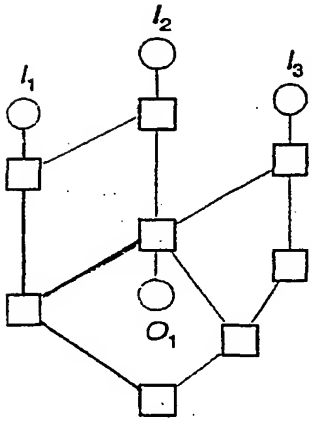



FIG. 23d

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 October 2001 (11.10.2001)

PCT

(10) International Publication Number
WO 01/75791 A3

(51) International Patent Classification[7]: G06N 3/12

(21) International Application Number: PCT/GB01/01517

(22) International Filing Date: 3 April 2001 (03.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0008291.7 4 April 2000 (04.04.2000) GB

(71) Applicant *(for all designated States except US)*: **ABER GENOMIC COMPUTING LIMITED** [GB/GB]: Llwyn yr Eos, Capel Bangor, Aberystwyth, Ceredigion SY23 3LZ (GB).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **KELL, Douglas, Bruce** [GB/GB]: Llechwedl Melyn, Pwgah, Aberystwyth, Ceredigion SY23 3NE (GB). **ROWLAND, Jeremy, John** [GB/GB]: Llwyn yr Eos, Capel Bangor, Aberystwyth, Ceredigion SY23 3LZ (GB). **GILBERT, Richard, James** [GB/GB]: Tan-y-Ffordd, Penuwch, Tregaron, Ceredigion SY25 6RA (GB).

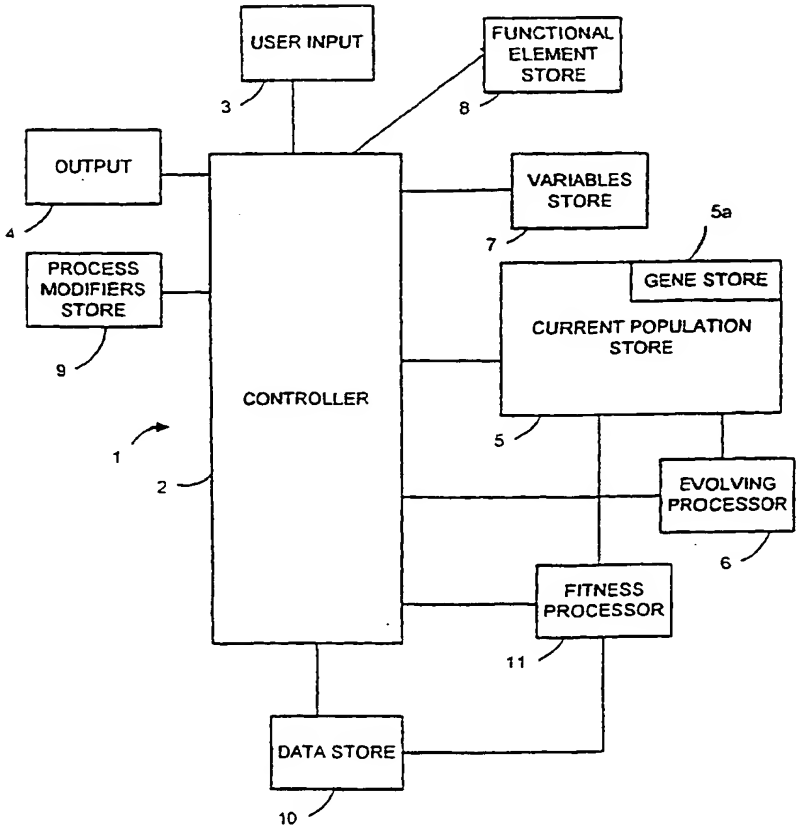
(74) Agents: **BERESFORD, Keith, Denis, Lewis** et al.: Beresford & Co., 2-5 Warwick Court, High Holborn, London WC1R 5DH (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian

*[Continued on next page]*

(54) Title: APPARATUS AND A METHOD FOR SOLVING PROBLEMS




(57) Abstract: A processor arrangement (2, 6, 11) is operable to provide an initial population of individuals each comprising a pluralify of structural elements each comprising a plurality of functional components defining operations to be performed on at least one input to the functional component and then to: (i) activate each individual to cause the operations defined by the functional components of that individual to be performed to produce a result; (ii) determine from each result a fitness of the corresponding individual for providing a mechanism or program for providing a solution to the problem; (iii) produce a new generation of individuals by causing an evolutionary process to occur involving at least one structural element and at least one functional component; and then to repeat (i) to (iii) until either an individual of the current generation represents a suitable way, structure, mechanism or program for providing a solution to the problem or a predetermined number of generations have been evolved.

WO 01/75791 A3

patent (AM. AZ. BY. KG. KZ. MD. RU. TJ. TM). European patent (AT. BE. CH. CY. DE. DK. ES. FI. FR. GB. GR. IE. IT. LU. MC. NL. PT. SE. TR). OAPI patent (BF. BJ. CF. CG. CI. CM. GA. GN. GW. ML. MR. NE. SN. TD. TG).

**Published:**

— *with international search report*

**(88) Date of publication of the international search report:**
20 June 2002

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*